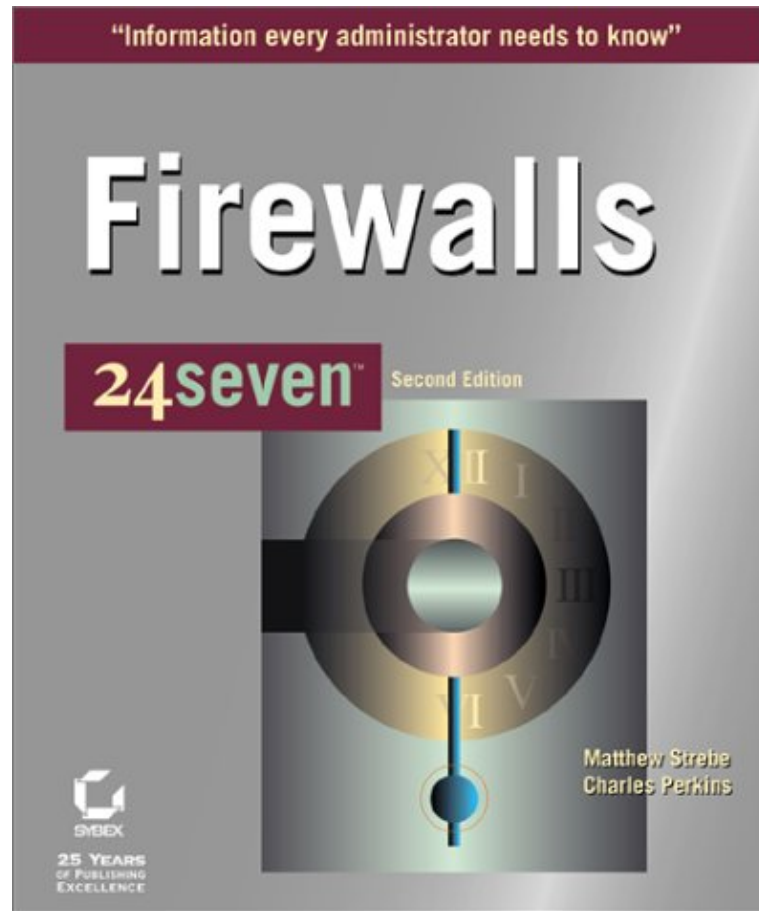


CEM



Firewalls 24Seven, Second Edition

Table of Contents

Firewalls 24Seven, Second Edition.....	1
Introduction.....	3
About This Book.....	3
How This Book is Organized.....	4
Part I: The Internet.....	4
Part II: Firewall Technology.....	4
Part III: Additional Security Tools.....	4
Part IV: Operating System Support for Firewalling.....	4
Part V: Commercial Firewalls.....	4
Where to Go From Here.....	4
Part I: The Internet.....	6
Chapter List.....	6
Part Overview.....	6
Chapter 1: Understanding Firewalls.....	7
Overview.....	7
Firewall Elements.....	7
Packet Filters.....	8
Network Address Translation.....	13
Proxies.....	13
Virtual Private Networks.....	15
Encrypted Authentication.....	16
Creating Effective Border Security.....	17
Comparing Firewall Functionality.....	18
Problems Firewalls Can't Solve.....	19
Border Security Options.....	21
Chapter 2: Hackers.....	28
Overview.....	28
Hacker Species.....	28
Security Experts.....	28
Script Kiddies.....	29
Underemployed Adult Hackers.....	30
Ideological Hackers.....	31
Criminal Hackers.....	31
Corporate Spies.....	32
Disgruntled Employees.....	33
Vectors of Attack.....	33
Physical Intrusion.....	34
Dial-up.....	34
Internet.....	35
Direct Connection.....	35
Hacking Techniques.....	36
Eavesdropping and Snooping.....	36
Denial of Service.....	41
Protocol Exploitation.....	44
Impersonation.....	46
Man-in-the-Middle.....	48

Table of Contents

Chapter 2: Hackers	
Hijacking.....	49
Chapter 3: TCP/IP from a Security Viewpoint.....	51
Overview.....	51
You Need to Be a TCP/IP Guru.....	51
TCP/IP Rules.....	52
The Bit Bucket Brigade.....	53
Layer 1: Physical.....	54
Layer 2: Data Link.....	58
Layer 3: Network.....	61
Chapter 4: Sockets and Services from a Security Point of View.....	77
Overview.....	77
Evaluating Socket-Based Services.....	77
How Complex Is the Service?.....	77
How Might the Service Be Abused?.....	78
What Information Does the Service Dispense?.....	78
How Much of a Dialog Does the Service Allow?.....	79
How Programmable or Configurable is the Service?.....	80
What Sort of Authentication Does the Service Use?.....	80
Your Network Profile.....	81
DNS, The Essential Service.....	81
Common Internet Services.....	86
Other Common Services.....	90
Windows-Specific Services.....	91
Standard Unix Services.....	92
Platform Neutral Services.....	94
Chapter 5: Encryption.....	98
Overview.....	98
How to Keep a Secret.....	98
Ciphers.....	98
Keeping Secrets Automatically.....	100
Keeping Secrets Electronically.....	100
Encryption in Your Network.....	102
Private Communications.....	103
Secure File Storage.....	104
User or Computer Authentication.....	104
Secure Password Exchange.....	105
A Conspiracy of Cryptographers.....	106
Algorithms.....	107
Symmetric Functions.....	108
Asymmetric Functions.....	109
Public Key Encryption.....	110
Protocols.....	111
Attacks on Ciphers and Cryptosystems.....	112
Digital Signatures.....	113
Steganography.....	114
Random Sequence Generation.....	114

Table of Contents

Part II: Firewall Technology.....	116
Chapter List.....	116
Part Overview.....	116
Chapter 6: Packet Filtering.....	117
Overview.....	117
How Stateless Packet Filters Work.....	117
Protocol Filtering.....	118
IP Address Filtering.....	118
TCP/UDP Ports.....	119
Filtering on Other Information.....	120
Problems with Stateless Packet Filters.....	121
OS Packet Filtering.....	122
How Stateful Inspection Packet Filters Work.....	122
Hacking through Packet Filters.....	124
TCP Can Only Be Filtered in 0th Fragments.....	125
Low Pass Blocking Filters Don't Catch High Port Connections.....	125
Public Services Must Be Forwarded.....	125
Internal NATs Can Defeat Filtering.....	126
Best Packet Filtering Practices.....	126
Use a Real Firewall.....	126
Disable All Ports By Default.....	126
Secure the Base OS.....	126
Chapter 7: Network Address Translation.....	128
Overview.....	128
NAT Explained.....	128
Translation Modes.....	131
Router Configuration for NAT.....	135
Problems with NAT.....	137
Hacking through NAT.....	138
Static Translation = No Security.....	138
Internal Host Seduction.....	139
The State Table Timeout Problem.....	139
Source Routing through NAT.....	140
Chapter 8: Application-Level Proxies.....	143
Overview.....	143
How Proxies Work.....	144
Security Advantages of Proxies.....	144
Performance Aspects of Proxies.....	148
Security Liabilities of Proxies.....	149
Performance Liabilities of Proxies.....	151
Explicit vs. Transparent Proxies.....	152
Proxy Best Practices.....	153
Use a Real Firewall.....	153
Disable Routing.....	154
Secure the Base Operating System.....	154
Disable External Access.....	155
Disable Excess Services.....	155

Table of Contents

Chapter 9: Virtual Private Networks.....	157
Overview.....	157
Virtual Private Networking Explained.....	157
IP Encapsulation.....	158
Cryptographic Authentication.....	160
Data Payload Encryption.....	160
Characteristics of VPNs.....	161
VPNs Are Cheaper Than WANs.....	161
VPNs Are Easier to Establish.....	162
VPNs Are Slower Than LANs.....	163
VPNs Are Less Reliable Than WANs.....	164
VPNs Are Less Secure Than Isolated LANs or WANs.....	165
Types of VPNs.....	166
Server-Based VPNs.....	166
Firewall-Based VPNs.....	167
Router-Based VPNs.....	168
VPN Architectures.....	168
Mesh VPNs.....	168
Hub and Spoke VPNs.....	169
Hybrid VPNs.....	169
Common VPN Implementations.....	170
IPSec.....	170
Layer 2 Tunneling Protocol (L2TP).....	173
PPTP.....	174
PPP/SSL or PPP/SSH.....	175
Secure Remote Access.....	176
VPN in the ISP.....	176
VPN in the Dial-Up Client.....	177
VPN Best Practices.....	177
Use a Real Firewall.....	178
Secure the Base Operating System.....	178
Use a Single ISP.....	178
Use Packet Filtering to Reject Unknown Hosts.....	178
Use Public-Key Encryption and Secure Authentication.....	179
Compress Before You Encrypt.....	179
Secure Remote Hosts.....	179
Prefer Compatible IPSec+IKE VPNs.....	179
 Chapter 10: The Ideal Firewall.....	 182
Overview.....	182
Defining Your Security Requirements.....	182
Home Offices.....	182
Small Service Businesses.....	184
Professional Firms.....	184
Manufacturers.....	184
Government Bureaus.....	185
Universities or Colleges.....	185
Internet Service Providers.....	185
Online Commerce Companies.....	186
Financial Institutions.....	186

Table of Contents

Chapter 10: The Ideal Firewall	
Hospitals.....	187
Military Organizations.....	187
Intelligence Agencies.....	187
Configuring the Rules.....	188
Rules about Rules.....	188
Rules for Security Levels.....	190
Aware.....	190
Concerned.....	191
Cautious.....	195
Strict.....	197
Paranoid.....	198
Chapter 11: Configuring a Real Firewall.....	200
The SonicWALL Appliance Wizard.....	200
SonicWALL Registration.....	208
SonicWALL Configuration.....	214
General.....	214
Log.....	216
Filters.....	218
Tools.....	222
Access.....	224
Advanced.....	228
DHCP.....	231
VPN.....	233
Anti-Virus.....	235
High Availability.....	236
Part III: Additional Security Tools.....	239
Chapter List.....	239
Part Overview.....	239
Chapter 12: Attack Profiles.....	240
Overview.....	240
Denial-of-Service Attacks.....	240
Ping of Death.....	240
Teardrop.....	241
UDP Floods.....	241
SYN Floods.....	242
Land Attack.....	243
Smurf Attack.....	243
Fraggle Attack.....	244
E-mail Bombs.....	244
Malformed Message Attacks.....	245
Exploitation Attacks.....	245
TCP/IP Connection Hijacking.....	245
Layer-2 Connection Hijacking.....	247
Password Guessing.....	248
Trojan Horses.....	249
Buffer Overruns.....	250

Table of Contents

Chapter 12: Attack Profiles

Information Gathering Attacks.....	250
Address Scanning.....	250
Port Scanning.....	251
Inverse Mapping.....	251
Slow Scanning.....	252
Architecture Probes.....	252
DNS Zone Transfers.....	253
Finger.....	253
LDAP.....	254
SNMP Leakage.....	254
Disinformation Attacks.....	254
DNS Cache Pollution.....	255
Registrar Usurpation.....	255
Forged E-mail.....	255

Chapter 13: Security Utilities.....258

Overview.....	258
Software You Already Have.....	258
Unix/Linux Utilities.....	258
IPChains/ipf.....	261
Windows Utilities.....	262
Cross Platform Tools.....	266
Security Analysis Tools.....	269
SATAN.....	269
WS-Ping.....	270
Internet Scanner.....	271
Protocol Analyzers.....	272
Sniffer Basic (Formerly NetXRay).....	272
Microsoft Network Monitor.....	273
CommView.....	273
TCPDump, IPTraf, and Snarf.....	273
Encryption Tools.....	274
Transparent Cryptographic File System.....	274
Encrypting File System (EFS).....	275
PGP.....	277
Scramdisk.....	277
Thawte Certificates.....	277
Password Strength Checkers.....	278
L0phtCrack.....	278
NetBIOS Auditing Tool.....	278
Personal Firewalls.....	279
BlackICE Defender.....	280
Norton Personal Firewall 2002.....	280
McAfee Firewall 3.0.....	281
CheckIt Firewall.....	281
Tiny Personal Firewall.....	281
ZoneAlarm.....	282

Table of Contents

Chapter 14: Intrusion Detection.....	283
Overview.....	283
Direct Intrusion.....	283
Intrusion Tools and Techniques.....	285
Intrusion Detection Systems.....	287
Inspection–Based Intrusion Detectors.....	287
Decoy Intrusion Detectors.....	288
Available IDS Systems.....	290
Windows System.....	290
NAI CyberCop.....	295
Tripwire.....	295
 Part IV: Operating Systems as Firewalls.....	 298
Chapter List.....	298
Part Overview.....	298
 Chapter 15: Windows as a Firewall.....	 299
Overview.....	299
Windows NT 4.....	299
Capabilities.....	300
Limitations.....	306
Windows 2000.....	307
CryptoAPI.....	308
Kerberos Authentication.....	308
Network Address Translation (NAT).....	310
Network Load Balancing.....	310
Improved Packet Filtering.....	311
IPX Packet Filtering.....	311
Layer–2 Tunneling Protocol (L2TP).....	311
IPSec.....	311
 Chapter 16: Open Source Firewalls.....	 314
Overview.....	314
Linux and IPChains or IPTables.....	314
Major Feature Set.....	315
Minor Feature Set.....	316
Security.....	316
Interface.....	317
Documentation.....	319
Cost and Support.....	319
The Trusted Information Systems Firewall Toolkit (TIS FWTK).....	319
Major Feature Set.....	320
Minor Feature Set.....	320
Security.....	320
Interface.....	321
Documentation.....	322
Cost and Support.....	323
FreeBSD and Drawbridge.....	323
Major Feature Set.....	323
Minor Feature Set.....	324

Table of Contents

Chapter 16: Open Source Firewalls

Security.....	324
Documentation.....	328
Cost and Support.....	328
OpenBSD and Ipf.....	329
Major Feature Set.....	329
Minor Feature Set.....	329
Security.....	330
Interface.....	330
Documentation.....	331
Cost and Support.....	331
Packet Filtering with DOS and IPRoute.....	332
Major Feature Set.....	332
Minor Feature Set.....	332
Security.....	333
Interface.....	333
Documentation.....	336
Cost and Support.....	336

Part V: Commercial Firewalls.....337

Chapter List.....	337
Part Overview.....	337

Chapter 17: Windows Firewalls.....338

Overview.....	338
Checkpoint Firewall-1.....	339
Major Feature Set.....	340
Minor Feature Set.....	341
Interface.....	342
Security.....	343
Documentation.....	343
Cost and Support.....	343
Symantec Enterprise Firewall.....	344
Major Feature Set.....	345
Minor Feature Set.....	346
Security.....	346
Interface.....	347
Documentation.....	348
Cost and Support.....	348
Microsoft Internet Security and Acceleration Server.....	348
Major Feature Set.....	349
Minor Feature Set.....	350
Security.....	352
Interface.....	353
Cost and Support.....	353

Chapter 18: Unix Firewalls.....355

Computer Associates eTrust Firewall.....	355
Major Feature Set.....	356
Minor Feature Set.....	356

Table of Contents

Chapter 18: Unix Firewalls

Interface.....	357
Security.....	357
Documentation, Cost, and Support.....	357
SecurIT Firewall.....	358
Major Feature Set.....	358
Minor Feature Set.....	359
Security.....	359
Documentation, Cost, and Support.....	360
NetWall.....	360
Major Feature Set.....	361
Minor Feature Set.....	361
Interface.....	362
Security.....	362
Documentation, Cost, and Support.....	362
Network Associates Gauntlet on the WebShield e–pliance.....	363
Major Feature Set.....	363
Minor Feature Set.....	365
Security.....	365
Interface.....	366
Documentation.....	367
Cost and Support.....	367
SunScreen Secure Net 3.1.....	367
Major Feature Set.....	367
Minor Feature Set.....	368
Interface.....	368
Security.....	369
Documentation, Cost, and Support.....	370

Chapter 19: Device and Specialty Firewalls.....372

Overview.....	372
SonicWALL.....	373
Major Feature Set.....	373
Minor Feature Set.....	374
Installation, Interface, and Documentation.....	374
Security.....	375
Cost and Support.....	375
WatchGuard Firebox 1000.....	376
Major Feature Set.....	376
Minor Feature Set.....	377
Installation.....	377
Security.....	377
Interface.....	378
Documentation.....	378
Cost and Support.....	378
Elron Firewall.....	379
Major Feature Set.....	380
Minor Feature Set.....	381
Interface.....	381
Security.....	382

Table of Contents

Chapter 19: Device and Specialty Firewalls

Documentation, Cost, and Support.....	382
GNAT Box.....	383
Major Feature Set.....	384
Minor Feature Set.....	385
Interface.....	385
Security.....	385
Documentation, Cost, and Support.....	386
BorderManager.....	386
Major Feature Set.....	386
Minor Feature Set.....	387
Interface.....	388
Security.....	388
Documentation, Cost, and Support.....	388
IBM Firewall for AS/400.....	389
Major Feature Set.....	390
Minor Feature Set.....	390
Interface.....	391
Security.....	391
Documentation, Cost, and Support.....	392
List of Figures.....	393
List of Tables.....	396
List of Tables.....	397
List of Sidebars.....	399

Firewalls 24Seven, Second Edition

Matthew Strebe
Charles Perkins



San Francisco London
Associate Publisher: Neil Edde

Acquisitions and Developmental Editor: Maureen Adams

Editor: Colleen Wheeler Strand

Production Editor: Liz Burke

Technical Editor: Sean Schluntz

Book Designer: Bill Gibson

Graphic Illustrator: Tony Jonick

Compositor: Nila Nichols

Proofreaders: Dave Nash, Laurie O'Connell, Jennifer Campbell, Yariv Rabinovitch, Nancy Riddiough, Emily Hsuan, Nanette Duffy

Indexer: Ted Laux

Cover Designer: Ingalls + Associates

Cover Illustrator: Hank Osuna

Copyright © 2002 SYBEX Inc., 1151 Marina Village Parkway, Alameda, CA 94501.

World rights reserved.

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way, including but not limited to photocopy, photograph, magnetic, or other record, without the prior agreement and written permission of the publisher.

First edition copyright © 2000 SYBEX Inc.

Library of Congress Card Number: 2001096982

ISBN: 0-7821-4054-8

SYBEX and the SYBEX logo are either registered trademarks or trademarks of SYBEX Inc. in the United States and/or other countries.

24seven and the 24seven logo are trademarks of SYBEX Inc.

Screen reproductions produced with FullShot 99. FullShot 99 © 1991–1999 Inbit Incorporated. All rights reserved.

FullShot is a trademark of Inbit Incorporated.

TRADEMARKS: SYBEX has attempted throughout this book to distinguish proprietary trademarks from descriptive terms by following the capitalization style used by the manufacturer.

The author and publisher have made their best efforts to prepare this book, and the content is based upon final release software whenever possible. Portions of the manuscript may be based upon pre-release versions supplied by software manufacturer(s). The author and the publisher make no representation or warranties of any kind with regard to the completeness or accuracy of the contents herein and accept no liability of any kind including but not limited to performance, merchantability, fitness for any particular purpose, or any losses or damages of any kind caused or alleged to be caused directly or indirectly from this book.

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

To Christy, always.
—Matt Strebe

To Joe.
—Charles Perkins

Acknowledgements

I'd like to thank my co-author, Charles Perkins, for handling all the Unix parts and for doing most of the firewall testing. I'd also like to thank Bryon Pinkston, Yuri Risovanny, Zach Little, Merrick Lozano, and my co-workers at Connetic, for giving me the time to write this stuff by handling the work I should have done.

I'd like to thank the people at Sybex for putting this book together, especially Maureen Adams for putting it together in the first place, Liz Burke and Colleen Strand who worked on this book daily, as well as Nila Nichols, Sean Schluntz, Dave Nash, Laurie O'Connell, Jennifer Campbell, Yariv Rabinovitch, Nancy Riddiough, Emily Hsuan, Nanette Duffy, and Ted Laux.

—Matthew Strebe

I'd like to thank everyone at Sybex for the hard work they've put into this book, especially Maureen Adams, Liz Burke, and Colleen Strand. I'd also like to thank my family for their constant support: Charles & Georgia, Donna & Cliff, Cathy & Jeff, Becky & Mike, and Joe.

—Charles Perkins

Introduction

Since the first edition of this book, firewalls have gone from esoteric and somewhat optional machines for the paranoid to mandatory guardians of the Internet required by just about everyone. Developed from rudimentary security systems that major computer vendors like Compaq and IBM created to secure their own networks in the mid-eighties, these network sentinels have developed in lock-step with the burgeoning threat of information warfare. Recently, there have been only incremental improvements in firewall technology, such as improvements in VPN interoperability between vendors. Much of the change in the firewall market has simply been a shakeout, where the strong firewalls have survived and the weak, expensive, and difficult to configure or buy have withered away.

The security problems of the past could be solved with simple packet filters and dial-back modem banks. The security problems of the future will require rifling through and validating every byte of an Internet message, requiring encrypted certification of a website's true identity before connecting, and then encrypting nearly everything that travels between. Fortunately, as technology and the technological society it mirrors progress, these measures will become simple and invisible. As vendors make operating systems more hardened against attack, the World Wide Web will secretly grow more secure for people who will freely surf the Web as they please, hampered only by the occasionally warning that a site is not accredited or that a message contains suspicious content. Linux already contains very strong built-in firewalling; and by the next edition of this book, Windows will be just as hardened—if Microsoft expects it to survive as an Internet service platform. This is as it should be.

The security problems of today are most effectively solved with firewalls and virtual private tunnels. Peripheral security utilities like intrusion detectors and security scanners do their part to alarm and alert, but firewalls will remain the foundation of Internet security until their functionality is built into the very protocols upon which the Internet operates and until every Internet-connected computer contains the equivalent of a firewall. Even then, centralized management of Internet policy may make firewalls a permanent addition to corporate networking.

About This Book

This book was written to accomplish one goal: to teach network administrators what they need to know to understand the Internet security threat, the technologies used to prevent it, and the products that exist to help them. It's the book I wish I'd had when I couldn't find a common language between various vendors that I could use to compare firewall literature and books heavy on theory to the marketing blurbs I read on websites, or when I needed help matching a specific customer's requirements to a specific firewall product.

This book will help you answer questions like these:

- What's the difference between packet filtering and stateful inspection, and why is it important?
- What's the difference between using Network Address Translation and a proxy server to hide clients?
- How much can I expect to budget for a firewall?
- Which firewall is right for my company?

This book was written primarily for active network administrators with the assumption that they understand the use and configuration of TCP/IP, who are used to working with Windows, or Unix

(although very little operating system–specific information is presented).

If you're not a network administrator, but you know you need a firewall, this book can still help you find one; a number of plug-and-play firewall devices exist that are both secure and easy to use and configure. If you fall into this later category, you may find your eyes glazing over during some of the more technical discussions early in the book. Feel free to skip over anything you don't understand and come back to it later if you need to.

How This Book is Organized

This book is divided into four parts that divide up 19 chapters. You should read through Parts I and II in order from beginning to end, but you can read the remainder of the book in any order.

Part I: The Internet

Chapters 1–5 cover information you should understand before we delve into firewall technology such as the Internet, the basic functions of firewalls, hackers, encryption, and details of the inner workings of TCP/IP.

Part II: Firewall Technology

Chapters 6–11 cover the five major technologies upon which most firewalls are based: packet filtering, Network Address Translation, proxying, authentication, and tunneling. It also details those measures you should take with any firewall to make sure it's securely configured, and walks through the configuration of a typical firewall.

Part III: Additional Security Tools

Chapters 12–14 detail security software and methods other than firewalls that you will use to enhance the security of your network.

Part IV: Operating System Support for Firewalling

Chapters 15 and 16 discuss what you can do with major operating systems to secure the services you provide. This is especially important for public servers.

Part V: Commercial Firewalls

Chapters 17–19 are the really unique part of this book—they provide an overview of a large portion of the commercially available firewall solutions. You can use these chapters to compare various firewalls and find the right fit for your organization.

Where to Go From Here

Security is not a static thing, it's a continually evolving process. You can't just plug in a firewall and expect it to solve your security problem forever. Attacks change, methods become obsolete, and so do firewalls. To obtain true security, you have to maintain constant vigilance. The easiest way I've found to do that is by getting on some of the mailing lists provided by firewall vendors and security

organizations like SANS and CERT and by visiting their websites (<http://www.sans.org/>, <http://www.cert.org/>).

Part I: The Internet

Chapter List

Chapter 1: Understanding Firewalls

Chapter 2: Hackers

Chapter 3: TCP/IP from a Security Viewpoint

Chapter 4: Sockets and Services from a Security Point of View

Chapter 5: Encryption

Part Overview

Topics Covered:

- How the Internet works
- How firewalls work
- Who hacks
- Hacker's motivations
- TCP/IP fundamentals
- TCP/IP higher level protocols
- How hackers exploit weaknesses in TCP/IP
- How encryption works
- How encryption provides security over the Internet
- How encryption provides a mechanism to prove user identity

Chapter 1: Understanding Firewalls

Overview

Nations without controlled borders cannot ensure the security and safety of their citizens, nor can they prevent piracy and theft. Networks without controlled access cannot ensure the security or privacy of stored data, nor can they keep network resources from being exploited by hackers.

The communication efficiency provided by the Internet has caused a rush to attach private networks directly to it. Direct Internet connections make it easy for hackers to exploit private network resources. Prior to the Internet, the only widely available way for a hacker to connect from home to a private network was by direct dialing with modems and the public telephony network. Remote access security was a relatively small issue.

When you connect your private network to the Internet, you are actually connecting your network directly to every other network that's attached to the Internet directly. There's no inherent central point of security control—in fact, there's no inherent security at all.

Firewalls are used to create security checkpoints at the boundaries of private networks. At these checkpoints, firewalls inspect all packets passing between the private network and the Internet and determine whether to pass or drop the packets depending on how they match the policy rules programmed into the firewall. If your firewall is properly configured, is capable of inspecting every protocol you allow to pass, and contains no serious exploitable bugs, your network will be as free from risk as possible.

There are literally hundreds of firewall products available, and there are different theories from different security experts on how firewalls should be used to secure your network. This chapter will explore the operation of a generic firewall in detail, outline the important features you need in a firewall, and discuss how firewalls should be deployed in networks of any size.

Firewall Elements

Firewalls keep your Internet connection as secure as possible by inspecting and then approving or rejecting each connection attempt made between your internal network and external networks like the Internet. Strong firewalls protect your network at all software layers—from the Data Link layer up through the Application layer.

Firewalls sit on the borders of your network, connected directly to the circuits that provide access to other networks. For that reason, firewalls are frequently referred to as border security. The concept of border security is important—without it, every host on your network would have to perform the functions of a firewall themselves, needlessly consuming computer resources and increasing the amount of time required to connect, authenticate, and encrypt data in local area, high-speed networks. Firewalls allow you to centralize all external security services in machines that are optimized for and dedicated to the task. Inspecting traffic at the border gateways also has the benefit of preventing hacking traffic from consuming the bandwidth on your internal network.

By their nature, firewalls create bottlenecks between the internal and external networks, because all traffic transiting between the internal network and the external must pass through a single point of control. This is a small price to pay for security. Since external leased-line connections are relatively slow compared to the speed of modern computers, the latency caused by firewalls can be

completely transparent. For most users, relatively inexpensive firewall devices are more than sufficient to keep up with a standard T1 connection to the Internet. For businesses and ISPs whose Internet traffic is far higher, a new breed of extremely high-speed (and high-cost) firewalls have been developed, which can keep up with even the most demanding private networks. Some countries actually censor the Internet using high-speed firewalls.

Firewalls function primarily by using three fundamental methods:

- **Packet Filtering** Rejects TCP/IP packets from unauthorized hosts and reject connection attempts to unauthorized services.
- **Network Address Translation (NAT)** Translates the IP addresses of internal hosts to hide them from outside monitoring. You may hear of NAT referred to as IP masquerading.
- **Proxy Services** Makes high-level application connections on behalf of internal hosts in order to completely break the network layer connection between internal and external hosts.

You can use devices or servers that perform only one of the above functions; for instance, you could have a router that performs packet filtering, and then a proxy server in a separate machine. This way, the packet filter must either pass traffic through to the proxy server, or the proxy server must sit outside your network without the protection of packet filtering. Both are more dangerous than using a single firewall product that performs all the security functions in one place. Most firewalls also perform two other important security services:

- **Encrypted Authentication** Allows users on the public network to prove their identity to the firewall, in order to gain access to the private network from external locations.
- **Virtual Private Networking** Establishes a secure connection between two private networks over a public medium like the Internet. This allows physically separated networks to use the Internet rather than leased-line connections to communicate. VPNs are also called encrypted tunnels.

Some firewalls also provide additional subscription-based services that are not strictly related to security, but which many users will find useful:

- **Virus Scanning** Searches inbound data streams for the signatures of viruses. Keeping up with current virus signatures requires a subscription to the virus update service provided by the firewall vendor.
- **Content Filtering** Allows you to block internal users from accessing certain types of content by category, such as pornography, hate-group propaganda, pornography, hacking information, and pornography. Keeping up with the current list of blocked sites for a specific category also requires a subscription.

Nearly all firewalls use these basic methods to provide a security service. There are literally hundreds of firewall products on the market now, all vying for your security dollar. Most are very strong products that vary only in superficial details. The remainder of this section covers the five primary functions that most firewalls support.

Packet Filters

The first Internet firewalls were simply packet filters, and packet filtering remains one of the key functions of today's firewalls. Filters compare network protocols (such as IP) and transport protocol packets (such as TCP) to a database of rules and forward only those packets that conform to the criteria specified in the database of rules. Filters can either be implemented in routers or in the TCP/IP stacks of servers (see Figure 1.1).

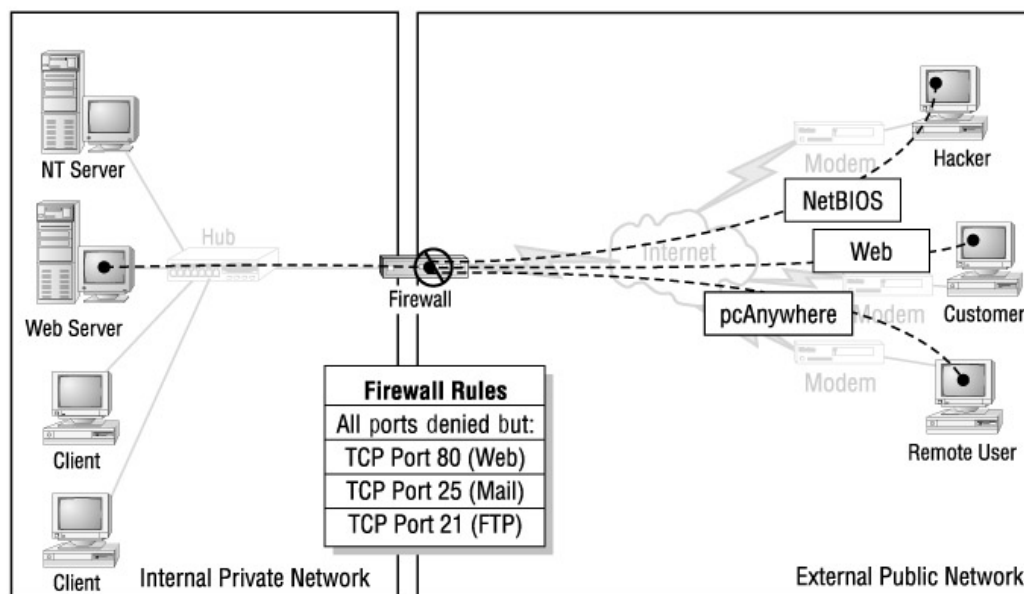


Figure 1.1: Filtered Internet connections block undesired traffic.

Filters implemented inside routers prevent suspicious traffic from reaching the destination network, whereas TCP/IP filter modules in servers merely prevent that specific machine from responding to suspicious traffic. The traffic still reaches the network and could target any machine on it. Filtered routers protect all the machines on the destination network from suspicious traffic. For that reason, filtering in the TCP/IP stacks of servers (such as that provided by Windows NT) should only be used in addition to router filtering, not instead of it.

Filters typically follow these rules:

- Drop inbound connection attempts but allow outbound connection attempts to pass.
- Eliminate TCP packets bound for those ports that shouldn't be available to the Internet (such as the NetBIOS session port) but allow packets that should be available (such as SMTP) to pass. Most filters can specify exactly which server a specific sort of traffic should go to—for instance, SMTP traffic on port 25 should only go to the IP address of a mail server.
- Restrict inbound access to certain IP ranges.

Warning Simple packet filters or routers with a packet filtering function that requires opening ports above 1023 for return channels are not effective security devices. These packet filters do not prevent internal users or Trojan horses from setting up a service on a client station in the port range above 1024 and simply listening for connection attempts from the outside. Firewalls (stateful inspection filters and security proxies) only open channels for servers that have been invited back in by a connection attempt from inside the security perimeter; choose them over simple packet filters that can't maintain the state of a connection.

Sophisticated filters examine the states of all connections that flow through them, looking for the telltale signs of hacking, such as source routing, ICMP redirection, and IP spoofing. Connections that exhibit these characteristics are dropped.

Internal clients are generally allowed to create connections to outside hosts, and external hosts are usually prevented from initiating connection attempts. When an internal host decides to initiate a TCP connection, it sends a TCP message to the IP address and port number of the public server (for example, <http://www.microsoft.com/> to connect to Microsoft's website). In the connection initiation message, it tells the remote server what its IP address is and on which port it is listening for a response (for example, localhost:2050).

The external server sends data back by transmitting it to the port given by the internal client. Since your firewall inspects all the traffic exchanged between both hosts, it knows that the connection was initiated by an internal host attached to its internal interface, what that host's IP address is, and on what port that host expects to receive return traffic. The firewall then remembers to allow the host addressed in the connection message to return traffic to the internal host's IP address only at the port specified.

When the hosts involved in the connection close down the TCP connection, the firewall removes the entry in its state table (its connection memory) that allows the remote host to return traffic to the internal host. If the internal host stops responding before closing the TCP connection (because, for example, it has crashed), or if the protocol in question does not support sessions (for example, UDP), the firewall will remove the entry in its state table after a programmed timeout of a few minutes.

Operating System Filtering

You might not be aware that most versions of UNIX and Windows include packet filtering in the TCP/IP protocol interface. You can use this filtering in addition to a strong firewall to control access to individual servers; you can also use this filtering to provide an additional measure of internal security inside your organization without the cost of a firewall. Just as filtering alone is not sufficient to protect your network entirely, your operating system's internal filtering is not sufficient to create a completely secure environment.

Security Limitations of Packet Filtering

Filtering does not completely solve the Internet security problem. First, the IP addresses of computers inside the filter are present in outbound traffic, which makes it somewhat easy to determine the type and number of Internet hosts inside a filter and to target attacks against those addresses. Filtering does not hide the identity of hosts inside the filter.

Additionally, filters cannot check all the fragments of an IP message based on higher-level protocols like TCP headers because the header exists only in the first fragment. Subsequent fragments have no header information and can only be compared to IP level rules, which are usually relaxed to allow some traffic through the filter. This allows bugs in the destination IP stacks of computers on the network to be exploited, and could allow communications with a Trojan horse installed inside the network. More modern true firewalls support rebuilding fragmented packets and then applying firewall rules to them.

Finally, filters are not complex enough to check the legitimacy of the protocols inside the network layer packets. For example, filters don't inspect the HTTP packets contained in TCP packets to determine if they contain exploits that target the web browser or web server on your end of the connection. Most modern hacking attempts are based upon exploiting these higher-level services because firewalls have nearly eliminated successful Network layer-hacking beyond the nuisance of denial-of-service attacks.

Variants of Windows

There are three major strains of Windows:

- 16-bit versions of Windows that run on top of MS-DOS including Windows 3.0, 3.1, and 3.11.
- 32-bit versions of Windows that run on MS-DOS including Windows 95, 98, and ME

- 32-bit versions of Windows that run on the NT Kernel, including NT 3.1, NT 3.5, NT 3.51, NT 4, 2000, and XP.

Throughout this book, when we use the term "Windows" we're talking about those versions based on the NT Kernel architecture unless we state otherwise.

Do not rely upon your operating system's built-in filtering alone to protect your network. You should use your operating system's filtering functions inside your network to establish filters to pass only those protocols you explicitly intend to serve. This prevents software from working in ways you don't expect and keeps Trojan horses from functioning even if they manage to get installed.

Basic OS filtering allows you to define acceptance criteria for each network adapter in your computer for incoming connections based on the following:

- IP protocol number
- TCP port number
- UDP port number

The filtering usually does not apply to outbound connections (those originating on your server), and is defined separately for each adapter in your system.

Note Windows 2000 supports outbound filtering; Windows NT 4 does not.

A typical server sets up services to listen on the following ports. These ports must be open through your filter in order for these services to work correctly.

Simple TCP/IP services usually listen on the following ports:

	Port	TCP/IP Service
7	Echo	
9	Discard	
13	Daytime	
17	Quote of the Day	
19	Character Generator	

Internet Servers usually listen on the following ports:

	Port	Server
21	File Transfer Protocol (FTP)	
23	Telnet	
70	Gopher	
80	World Wide Web (HTTP)	
119	Net News (NNTP)	
22	Secure Shell	
443	Secure HTTP (HTTPS)	

File Servers usually listen on the following ports:

Port	Service
53	Domain Name Service (DNS service, if installed)
135	RPC Locator Service (Windows NT only)
137	NetBIOS Name Service (WINS servers only)
139	NetBIOS Session Service (Windows network and SMB/CIFS servers only)
515	LPR is used by the TCP/IP print service, if installed.
530	Remote Procedure Call (RPC connections are used by the Windows NT WinLogon service as well as many other high-level network applications.)
3389	Windows Terminal Services accepts connections on this port using the RDP protocol

Mail Servers are usually configured to listen on the following ports:

Port	Mail Server
25	Simple Mail Transfer Protocol (Mail server to server exchanges)
110	Post Office Protocol version 3 (Server to client mail exchanges)
143	Internet Mail Access Protocol (Client access to mail server)

If you install other service software, you must make sure your server's filter is set up to listen on the ports required by the service—otherwise the service will not work. Find out from the software manufacturer which ports are required for that service. This does not apply to border firewalls, which should only be configured to pass a service if you intend to provide that service to the public.

General Rules for Packet Filtering

There are two basic approaches you can take to security: Pessimistic, where you disable all access except that which you know is necessary, and optimistic, where you allow all traffic except that which you know is harmful. For security purposes, you should always take a pessimistic approach, because the optimistic approach presumes that you know every possible threat in advance, which is not possible. Consider the following general guidelines when you use packet filtering:

- Disallow all protocols and addresses by default, and then explicitly allow services and hosts you wish to support.
- Disallow all connection attempts to hosts inside your network. By allowing any inbound connections, you allow hackers to establish connections to Trojan horses or exploit bugs in service software.
- Filter out and do not respond to ICMP redirect and echo (ping) messages. Drop all packets that are TCP source routed. Source routing is rarely used for legitimate purposes.
- Drop all external routing protocol (RIP, OSPF) updates bound for internal routers. No one outside your network should be transmitting RIP updates.
- Consider disallowing fragments beyond number zero, since this functionality is largely obsolete and often exploited.
- Place public service hosts like web servers and SMTP servers outside your packet filters rather than opening holes through your packet filters.
- Do not rely upon packet filtering alone to protect your network.

Network Address Translation

Network Address Translation (NAT) solves the problem of hiding internal hosts. NAT is actually a network layer proxy: A single host makes requests on behalf of all internal hosts, thus hiding their identity from the public network. Windows 2000 and XP, Linux, and many modern UNIX operating systems provide this function as part of the operating system distribution. Windows NT does not.

NAT hides internal IP addresses by converting all internal host addresses to the address of the firewall. The firewall then retransmits the data payload of the internal host from its own address using the TCP port number to keep track of which connections on the public side map to which hosts on the private side. To the Internet, all the traffic on your network appears to be coming from one extremely busy computer.

NAT effectively hides all TCP/IP-level information about your internal hosts from prying eyes on the Internet. Address translation also allows you to use any IP address range you want on your internal network even if those addresses are already in use elsewhere on the Internet. This means you don't have to request a large block of IP addresses from ARIN or reassign network numbers from those you simply plugged in before you connected your network to the Internet.

Warning Although you can use any block of IP addresses behind a firewall with NAT, be aware that you may encounter strange problems accessing Internet hosts that have the same public IP address as a computer inside your network. For that reason, use the reserved 192.168.0.0 network or the 10.0.0.0 network inside your firewall to avoid these problems.

Finally, NAT allows you to multiplex a single public IP address across an entire network. Many small companies rely upon the services of an upstream Internet service provider that may be reluctant to provide large blocks of addresses because their own range is relatively restricted. You may want to share a single dial-up or cable modem address without telling your ISP. These options are all possible using network address translation.

On the down side, NAT is implemented only at the TCP/IP level. This means that information hidden in the data payload of TCP/IP traffic could be transmitted to a higher-level service and used to exploit weaknesses in higher-level traffic or to communicate with a Trojan horse. You'll still have to use a higher-level service like a proxy to prevent higher-level service security breaches.

Additionally, many protocols also include the host's IP address in the data payload, so when the address is rewritten while passing through the NAT, the address in the payload becomes invalid. This occurs with active-mode FTP, H.323, IPsec, and nearly every other protocol that relies upon establishing a secondary communication stream between the client and the server.

NAT is also a problem for network administrators who may want to connect to clients behind the NAT for administrative purposes. Because the NAT has only one IP address, there's no way to specify which internal client you want to reach. This keeps hackers from connecting to internal clients, but it also keeps legitimate users at bay as well. Fortunately, most modern NAT implementations allow you to create port-forwarding rules that allow internal hosts to be reached.

Proxies

NAT solves many of the problems associated with direct Internet connections, but it still doesn't completely restrict the flow of packets through your firewall. It's possible for someone with a network monitor to watch traffic coming out of your firewall and determine that the firewall is translating addresses for other machines. It is then possible for a hacker to hijack TCP connections or to spoof

connections back through the firewall.

Application-level proxies prevent this. They allow you to completely disconnect the flow of network-level protocols through your firewall and restrict traffic only to higher-level protocols like HTTP, FTP, and SMTP. Application-level proxies are a combination of a server and a client for the specific protocol in question. For example, a web proxy is a combination of a web server and a web client. The protocol server side of the proxy accepts connections from clients on the internal network, and the protocol client side of the proxy connects to the public server. When the client side of the proxy receives data from the public server, the server side of the proxy application sends it to the ultimate inside client. Figure 1.2 shows exactly how this works.

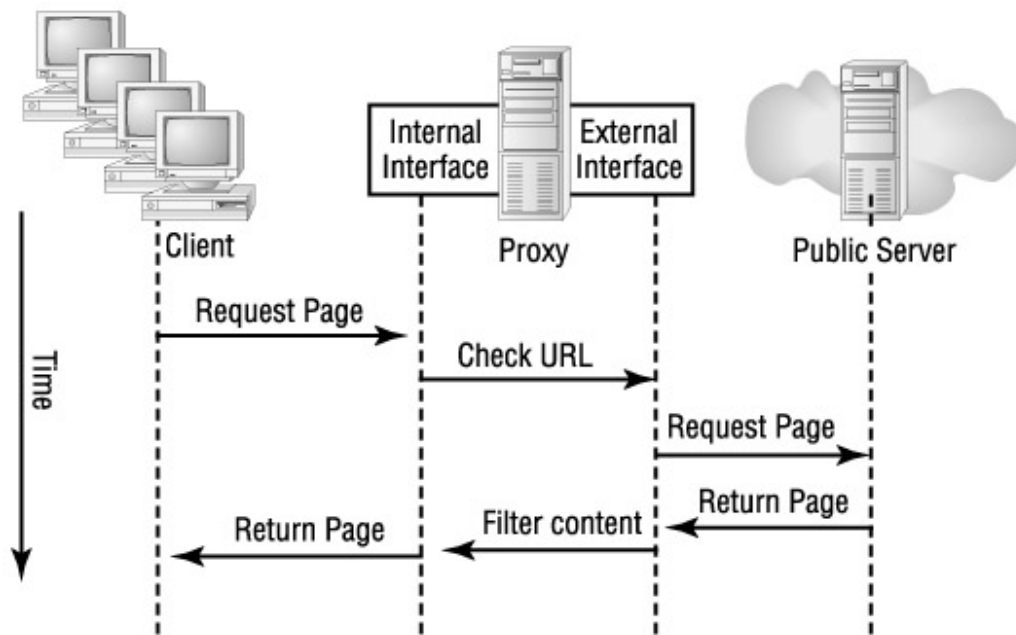


Figure 1.2: Proxy servers receive requests on the private network and regenerate them on the public network.

Proxies straddle two networks that are not connected by routers. When a client on the protected network makes a connection to a server on the public side, the proxy receives the connection request and then makes the connection on behalf of the protected client. The proxy then forwards the response from the public server onto the internal network. Proxies essentially perform a benign man-in-the-middle attack, and they provide a good example of how any intermediate system between you and another end system could potentially perform a more malicious sort of processing without your permission.

Application proxies (like Microsoft Proxy Server) are unlike Network Address Translators and filters in that the Internet client application is (usually) set up to talk to the proxy. For instance, you tell Internet Explorer the address of your web proxy, and Internet Explorer sends all web requests to that server rather than resolving the IP address and establishing a connection directly.

Application proxies don't have to run on firewalls; any server, either inside or outside your network, can perform the role of a proxy. Without a firewall, you still don't have any real security, so you need both. At least some sort of packet filter must be in place to protect the proxy server from network layer denial-of-service attacks (like the infamous "ping of death"). And, if the proxy doesn't run on the firewall, you'll have to open a channel through your firewall one way or another. Ideally, your firewall should perform the proxy function. This keeps packets from the public side from being forwarded through your firewall.

Some firewall proxies are more sophisticated than others. Some have the functionality of an IP filter and masquerade, so they can simply block outbound connection attempts (on port 80 in the case of HTTP) to remote hosts rather than having the client software configured to address the proxy service specifically. The firewall proxy then connects to the remote server and requests data on behalf of the blocked client. The retrieved data is returned to the requesting client using the firewall's NAT functionality in order to look just like the actual remote server. Proxies that operate in this manner are said to be transparent.

Security proxies are even capable of performing application-level filtering for specific content. For instance, some firewall HTTP proxies look for tags in HTML pages that refer to Java or ActiveX embedded applets and then strip out that content from them. This prevents the applet from executing on your client computers and eliminates the risk that a user will accidentally download a Trojan horse. This sort of filtering is extremely important because filtering, proxying, and masquerading can't prevent your network from being compromised if your users are lured into downloading a Trojan horse embedded in an ActiveX applet.

You may have noticed that as we climb through the networking layers, the security services have gotten more specific. For instance, filtering is specific to IP and then to TCP and UDP. Applications that use IP with other protocols like Banyan Vines must use special high-cost or unusually robust firewalls.

Proxies are extremely specific because they can only work for a specific application. For instance, you must have a proxy software module for HTTP, another proxy module for FTP, and another module for Telnet. As these protocols evolve (HTTP is particularly fast moving), the proxy module for that protocol will have to be updated.

Many protocols are either proprietary or rare enough that no security proxies exist. Proxies don't exist for proprietary application protocols like Lotus Notes, so those protocols must either be sent through a Network layer filter or be proxied by a generic TCP proxy that regenerates the packet but simply transfers the payload. SOCKS is a specific form of generic proxy, which are sometimes called circuit-level gateways. Although generic proxying cannot prevent attacks from the content of a protocol, it is still more secure than filtered routing because the Network layer packets are completely regenerated and thus scrubbed of malformations that might not be detected by the firewall.

In many cases, you "roll-your-own" proxy by using a combination of the protocol server and the protocol's client on the same machine. For example, say you've got a network that is disconnected from the Internet, but a Windows server has two network interfaces, one on the Internet and one on the private network. If you use the Terminal Services functionality of Windows 2000 to attach to the server on its public side, you can then run a Terminal Services client on that machine to reach a machine on the interior of the network. In practice, this actually works a lot better than you might presume, although it's not a particularly good security practice.

Whenever possible, use proxy servers for all application protocols. Consider disallowing services for which you do not have proxy servers. Use high-level proxies capable of stripping executable content, like ActiveX and Java, from web pages.

Virtual Private Networks

Virtual Private Networks (VPNs), also called encrypted tunnels, allow you to securely connect two physically separated networks over the Internet without exposing your data to viewing by unauthorized intermediate parties. VPNs by themselves could be subject to redirection attempts,

spoofed connection initiation, and all manner of hacking indignity while the tunnel is being established. But when implemented as an integral part of a firewall, the firewall authentication and security services can be used to prevent exploitation while the tunnel is being established.

Once established, VPNs are impervious to exploitation so long as the encryption remains secure. And, since firewalls sit at the Internet borders, they exist at the perfect terminal points for each end of the tunnel. Essentially, your private networks can pass traffic as if they were two subnets in the same domain.

VPNs also allow users to address remote internal hosts directly by their hidden IP addresses; Network Address Translators and packet filters would prevent this if the connection attempt came directly from the Internet.

Tip The Point-to-Point Tunneling Protocol for Windows NT provides an encrypted tunnel using the security services of the Remote Access Server. Windows 2000 provides support for the more modern Layer-2 Tunneling Protocol (L2TP) and IP Security (IPSec) in transport mode. Most distributions of Linux include support for encrypted tunnels, such as the Point-to-Point Protocol (PPP) over Secure Socket Layer (SSL).

Use leased lines rather than VPNs whenever it is cost effective. Use VPNs for all communications over the Internet between organizational units when leased lines are not available or are cost prohibitive. If you are using VPNs as your primary connection method between organizational units, you'll have far better performance if you use the same ISP at every site, because the VPN traffic won't have to be routed through the congested commercial Internet exchanges. Never communicate private information between organizational units over the Internet without using some form of encryption. Unencrypted packet headers contain valuable nuggets of information about the structure of your internal network.

Note Technically, leased lines are not guaranteed to be secure either, but they are free of Internet hackers. If you need to secure your data from the possibility of government wiretaps or serious corporate espionage, you should use a VPN over leased lines as well.

Encrypted Authentication

Encrypted authentication allows external users on the Internet to prove to a firewall that they are authorized users and thereby authorized to open a connection through the firewall to the internal network. The encrypted authentication might use any number of secure authentication protocols. Once the connection is established, it may or may not be encrypted, depending upon the firewall product in use and whether additional software has been installed on the client to support tunneling.

Using encryption authentication is convenient because it occurs at the transport level between a client software package and the firewall. Once the connection is open, all normal application software and operating system logon software will run without hindrance—so you don't have to use special software packages that support your specific firewall.

Unfortunately, encrypted authentication reduces the security of your firewall. By its nature, it causes the following problems:

- The firewall must respond on some port because it listens for connection attempts. This can show hackers that the firewall exists.
- The connection could be redirected using ICMP after establishment, especially if it's not encrypted.

- A hacker who monitored the establishment might be able to spoof the address of the authorized client to gain access inside the network without redirecting any existing connections.
- A stolen laptop computer with the appropriate keys could be used to gain access to the network.
- Work-at-home employees could become a target for breaking and entering because their computers are able to access the private network.
- The authentication procedure could be buggy or less than completely secure, thus allowing anyone on the Internet to open holes through the firewall.

Each of these risks is less than likely to actually occur. Administrators of medium-to low-risk environments should not feel uncomfortable using encrypted authentication as long as the connection is encrypted for the duration.

Creating Effective Border Security

To maintain the absolute minimum level of effective Internet security, you must control your border security using firewalls that perform all three of the basic firewall functions (packet filtering, Network Address Translation, and high-level service proxying). Your firewalls must also be dedicated primarily to the performance of firewall functions; avoid the temptation to run other services such as mail, web, or other public services on the firewall unless the service software comes from the firewall software vendor. Even in this case, be aware that you are increasing your risk, because a bug in any of the high-level services running on your firewall might be exploited to bypass the firewall completely. This recommendation is not as theoretical as it sounds: Unix Sendmail is notorious for the number of buffer-overflow attacks it has been susceptible to, as is Internet Information Services, the Windows web server. If these services run on your firewall, it can be compromised easily.

Again, simply minimize the services running on the firewalls. This reduces the complexity of the software running on the machine, thereby reducing the probability that a bug in the operating system or security software will allow a security breach. In the case of Windows, very few of the services in the service Control Panel are needed for a computer running only as a firewall. Turn off all services that the server will allow you to shut off and set them to start manually. In the case of Linux, install only those packages necessary for the operation of the firewall, or select the "firewall" installation option if the distribution has one. Normally, you won't have to deal with this because the firewall software installation program will shut down all unnecessary services for you. If it doesn't, look elsewhere for firewall software.

It's always tempting to pile services like HTTP, FTP, Telnet, Gopher, and mail onto the same machine you use as an Internet router and firewall because it's cheaper and because that machine probably has a lot of spare compute time and disk space. Unfortunately, few operating systems are both secure enough and bug-free enough to guarantee that services won't interfere with each other or that a service won't crash the firewall. It's also quite probable that a high-level service running on the firewall, even if it doesn't affect other security services, could provide a way to circumvent the security services of the firewall. And lastly, as I mentioned earlier in this chapter, many services contain logon banners or automatically generated error pages that identify the firewall product you are using. This could be dangerous if hackers have found a weakness in your specific firewall. You want to make it difficult to determine which operating system your firewall is running.

You must also enforce a single point of control in your firewall policy. If you have more than one firewall in your company (perhaps one firewall attaching each remote office to the Internet), you

need to make absolutely certain they are all configured the same way. Enterprise firewall management software features aid in this endeavor.

Warning A lapse on any of your firewalls can compromise your entire network, especially if you use secure tunneling or private leased lines to connect offices. Hackers can be relied upon to use the path of least resistance.

Comparing Firewall Functionality

There is a common misconception among network administrators that a firewall has to be based on the same operating system as the network file servers—Unix firewalls for Unix-based networks and NT firewalls for Windows NT-based networks. In fact, there's no functional reason why the operating system used by a firewall should be the same as that used by the network, since (and only in very special circumstances) you'll never run any other software on the firewall computer. In fact, these days, most firewalls come as preconfigured computers running a completely proprietary operating system.

All firewalls filter TCP/IP traffic, and in most cases you'll set them up once and leave them to do their job, with minor tweaks as security policies and work habits change in the organization. Some firewalls run proprietary operating systems that aren't related to Unix or Windows at all; they are just as appropriate on any network.

The second most important factor in choosing a firewall operating system (after security, of course) is familiarity—the administrator should be familiar with the user interface and know how to configure the firewall correctly. Most Windows-based firewalls are easier to set up than Unix-based firewalls, but many Unix-based firewalls are catching up by using Java or web-based graphical interfaces that run remotely on the administrator's PC.

Some firewall vendors claim that their products are superior to firewalls based on Windows or standard versions of Unix because the products are based on a "hardened" implementation of the TCP/IP protocol stack or a theoretically more secure operating system. They also claim that bugs in Windows NT or Unix releases can be exploited to get past the firewall software of their competitors. While this may be true, those vendors can't prove that similar bugs don't exist in their own software. In fact, there's no practical way to prove that complex code is bug free, and firewall vendors are no more likely to get it absolutely right than are large vendors like Microsoft or Sun.

One major advantage of using a widely available operating system as the foundation of a firewall is that the code is put through its paces by millions of users. Bugs are more likely to be found and corrected, and patches are available far sooner and with greater regularity than is true for proprietary products provided by smaller vendors who usually don't have the programming resources to throw at problems as they arise. On the down side, common operating systems are subject to far more hacking attempts than uncommon ones; Windows bears the brunt of hacking attempts because it's the most common operating system and because hackers hate Microsoft. For this reason, it's the most often compromised operating system, although Unix (including Linux) isn't theoretically any more secure.

Many firewall products that are based on a standard operating system don't rely on the standard TCP/IP stack or higher-level services that ship with the operating system; they implement their own TCP/IP stack so that they can have absolute control over its operation. The base operating system serves only as a platform for the firewall software, providing functions like booting, multitasking, and user interface.

Firewall products vary in the following ways:

- **Security** Some firewall products are fundamentally flawed because they rely too heavily on the host operating system, because they contain bugs that can be exploited, or because there is a flaw in the authentication protocol used for remote authentication.
- **Interface** Some firewalls are very difficult to configure because you must administer them via Telnet or an attached console and learn some cryptic command-line interface. Others use very intuitive graphical interfaces that make configuration easy and obvious (well, obvious to us geeks, anyway).
- **Enterprise Functionality** Some firewalls are fortresses unto themselves, while others use a centrally maintained security policy that is replicated among all firewalls in the enterprise.
- **Security Features** Many firewalls offer important security features such as virtual private networking and encrypted authentication to allow remote office networking with a high degree of security. In many firewalls, VPN is an extra-cost feature that must be enabled by purchasing an additional license.
- **Service Features** Some firewalls include services such as FTP, Telnet, HTTP, and so forth so that you don't have to dedicate a machine to those functions. These features can be convenient, but they're often somewhat obsolete in functionality and can reduce the security of the firewall if they aren't properly implemented. Also, many services reveal a copyright that tells hackers exactly which firewall product you are using and allows them to target any weaknesses it may have.

Your primary criterion for firewalls should be security. The next most important feature is ease of use for you; you must be able to correctly configure a firewall for it to work correctly. Flashy features, performance, and services galore are tertiary considerations after the key issues of security and ease of use.

Problems Firewalls Can't Solve

No network attached to the Internet can be made completely secure. Firewalls are extremely effective and they will keep the hacking masses at bay, but there are so many different ways to exploit network connections that no method is entirely secure. Many administrators mistakenly assume that once their firewall is online and shown to be effective, their security problem is gone. That's simply not the case.

For example, let's say that the only thing you allow through your firewall is e-mail. An employee gets a message from a branch office asking him to e-mail a CAD file to them. So the employee looks at the From address, verifies that it's correct, clicks reply, attaches the file, and unknowingly sends the CAD file to the hackers who forged the e-mail request because the Reply-to address isn't the same as the From address. Your firewall can't realistically do anything about this type of exploitation because many typical users have different From and Reply-to addresses for very valid reasons, (like they send mail from multiple e-mail addresses but only want to receive mail at one0.

Another problem firewalls can't solve is protection against protocols you decide to allow. For example, if you have a Windows-based IIS public web server on your network, your firewall must forward port 80 to it. Hackers can then attach to the web server as if they were typical web browsers and then exploit the hundreds of known bugs in IIS to gain remote administrative access to it. Once they have control of your web server, they're "inside" your network, and can use that web server to proxy an attack to the interior of your network, unless you have additional firewall policy preventing it.

There is another serious threat to the security of your network: hidden border crossings. Modems

provide the ability for any user on your network to dial out to their own Internet service provider and completely circumvent your firewall. Modems are cheap and they come in most computers sold these days. All modern client operating systems come with the software required for setting up modems to connect to a dial-up Internet service provider. And it's a good bet that most of your computer-savvy employees have their own dial-up networking accounts they could use from work.

Most users don't understand that all IP connections are a security risk. Modem PPP connections to the Internet are bidirectional just like leased lines. And there's a good chance that their client has file sharing turned on, so their computer can be exploited directly from the Internet.

Warning It's quite common for businesses with firewalls to allow unrestricted file and print sharing among peers because it's an easy and efficient way for users to transfer files. If one of those users is dialed into the Net, it's also an easy and efficient way for hackers to transfer your files. Remember that the AOL dialer provides PPP service, so it's not any more secure than any other dial-up ISP.

Why would a user choose a dial-up modem connection when they have a fast and secure Internet connection? Reasons might include the following:

- Your firewall doesn't pass Internet Relay Chat and they want to talk to their friends.
- So they can use NetPhone to talk to their mother for free.
- So they can work from home using pcAnywhere.
- Because AOL uses a port your firewall doesn't pass and they want to check their personal e-mail.
- Because you filter FTP and they want to download a file.
- Because your network is configured to block pornography sites.

Users dial out so they can circumvent your security policy without your knowledge. To control border security, you must control all the border crossings; it must be impossible to establish a new border crossing without your permission. Exceptions to this rule endanger the security of your entire network.

Reinforcing the Borders

Here are some tips for taking control of your border crossings:

- Reduce the number of connections to the Internet to the minimum number possible: one per campus. Many large organizations allow only a single link to the Internet at headquarters and then route all remote offices to that point using the same frame relay lines used to connect internal networks. Even if you use VPN to connect your remote offices, consider requiring them to route through your central firewall to reach the Internet—this way, you can control firewall policy on a single machine.
- Don't allow dial-up connections to the Internet. Remove modems and all other uncontrolled network access devices. Disable free COM ports in the BIOS settings of client computers and password protect the BIOS to prevent users from overriding your security settings.
- Don't allow unrestricted file sharing. Use file sharing with user-based authentication or, at the very least, with passwords. Don't install file and print sharing on client computers unless absolutely necessary. Encourage users to store all files on network file servers, and create server pools of resources like CD-ROMs or modems that can be centrally controlled.
- Configure internal client computers with IP addresses in the 192.168.0.0 or the 10.0.0.0 domains, which are not routed over the Internet. Use NAT to translate these internal addresses to routable external addresses on your firewall. This may prevent hackers from

exploiting modem connections into your network beyond the computer that established the connection.

Border Security Options

Once you've got your firewall running on the border between your private network and the Internet, you're going to run into a problem. How do you provide the public services your customers need while securing your internal network from attack? There is more than one answer to this question, and which one is right depends entirely upon your security posture and the level of service you need to provide.

Methods used by companies to protect their networks range from the simple to the complex, the risky to the very secure. Such methods include the following (in order of security risk from highest to lowest):

1. Filtered packet services
2. Single firewall with internal public servers
3. Single firewall with external public servers
4. Dual firewalls or DMZ firewalls
5. Enterprise firewalls
6. Disconnection

The following sections discuss each method in detail, along with relative risks and issues.

Filtered Packet Services

Most Internet service providers provide packet filtering as a value-added service for leased-line customers. For a small monthly charge (generally about \$100), your ISP will set up their firewall to filter traffic into and out of your network. Some ISPs also offer proxy servers and NAT, but you may still be at risk from security attacks by other customers served by that ISP. Remember that all hackers have ISPs too. Figure 1.3 illustrates how filtered packet services work.

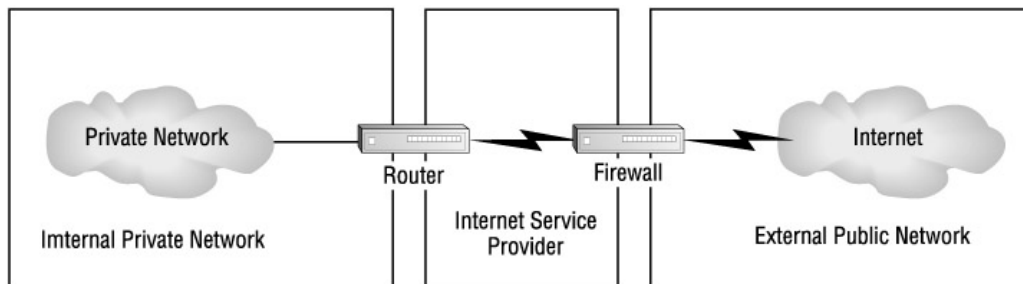


Figure 1.3: Filtered packet service

There are a number of problems with filtered firewall services:

- Packet filters can be exploited more easily than complete firewalls.
- Your security is in the hands of a third party. Their motivations may not always coincide with yours, especially if a legal dispute arises between your company and theirs.
- The responsibility for reliability isn't controllable.
- It's not in the best interest of the ISP to alert you that there has been a compromise.
- There's rarely any provision for alarming and alerting.
- Configuration is a difficult and error-prone administrative hassle. Reconfiguration is also a pain in the neck if the ISP doesn't have a strong customer support ethic.

- You are probably vulnerable to the ISP's other subscribers, who are usually inside the same firewall.

ISP–provided packet filters have the following advantage:

- No up–front capital expenditure is required.

Even if the firewall service provided by an ISP were complete, it would still never be a good idea to put the security of your network in the hands of another organization. You don't know anything about your ISP's employees, and you don't know what measures your ISP might take if for some reason a dispute arose between your company and theirs. Add to that these simple facts: most people who can hack do so at least occasionally and many good hackers work for the people who can get them closest to the action.

Locally control and administer all security services for your network. Don't put responsibility for the security of your network in the hands of an external organization. Don't rely solely on packet filters for security protection from the Internet.

The Single–Firewall Approach

The simplest complete border security solution is that of the single firewall. With one firewall and one connection to the Internet, you have a single point of management and control. Figure 1.4 shows a single firewall border security solution.

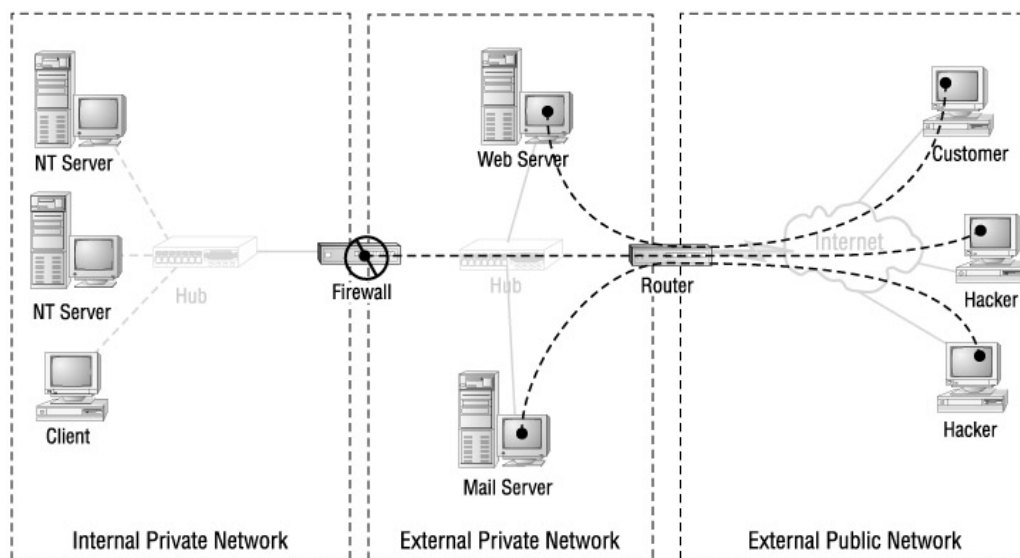


Figure 1.4: A single firewall with public servers exposed to the Internet

You have a problem if you intend to provide public services like an FTP site or website, or if you want to operate a mail server. You must either open a connection through your firewall to an internal host, or you must expose your public server to the Internet without the protection of a firewall. Both methods are risky.

The problem with putting public servers, like mail servers, outside your firewall is that they are at risk for unrestricted hacking. You can set these computers up so that they don't contain much useful information, but hacking attempts could easily cause denial of service if your servers are crashed, or at least cause embarrassment if hackers modify your web pages. Figure 1.5 shows public servers inside the firewall.

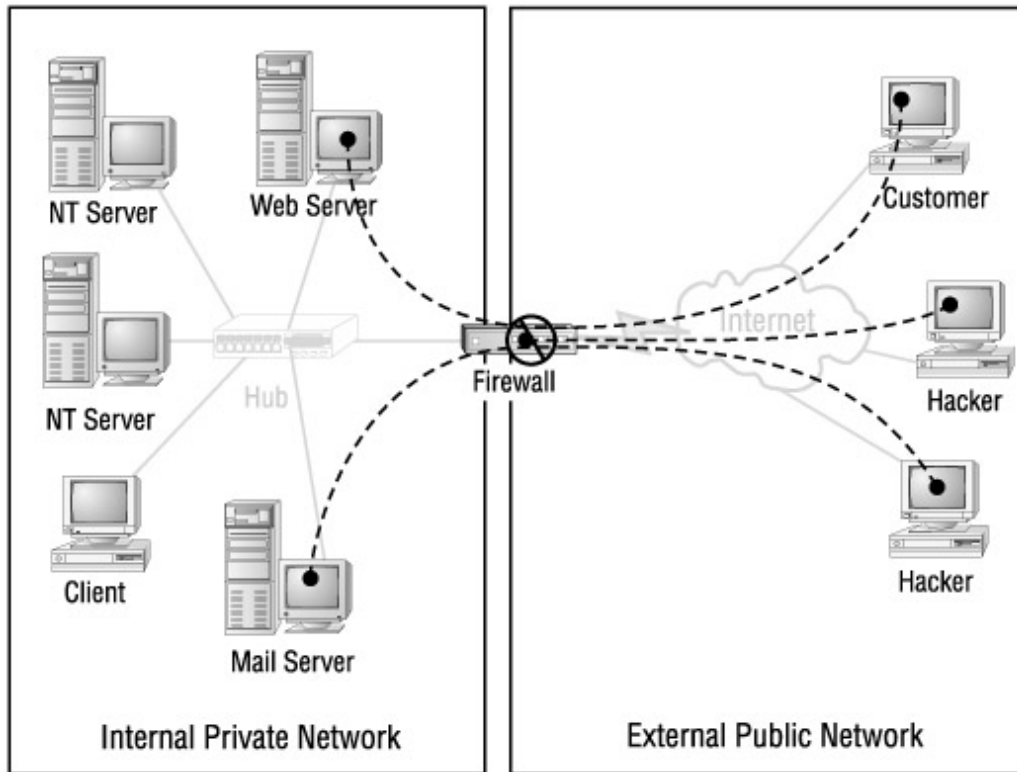


Figure 1.5: A single firewall with public servers protected but allowing external traffic in through the firewall

The problem with opening a path through your firewall for externally sourced connection attempts is that inappropriate packets could potentially make their way onto your internal network if they look like packets that conform to the rules used by your packet filter. It also means that a hacker who manages to exploit a bug in high-level service software might gain control of a computer inside your network—a very dangerous situation. For this reason, most organizations put public servers outside their firewalls and simply do not allow any external connections in through the firewall.

Dual Firewalls and Demilitarized Zones

You can reduce the risk of having exposed public servers with two firewalls and two levels of firewall protection. Basically, you put the first firewall at your Internet connection and secure your web servers behind it. It provides strong security, but allows connection attempts from the Internet for the services you want to provide.

Between that network and your internal network, you place a second firewall with a stronger security policy that simply does not allow external connection attempts and hides the identity of internal clients. Figure 1.6 shows a network with two firewalls providing two levels of security.

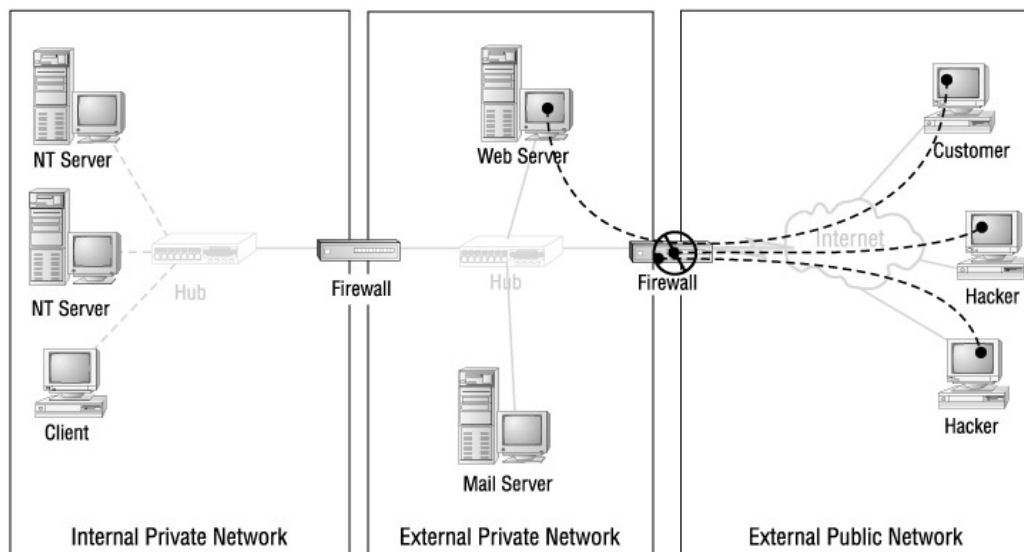


Figure 1.6: Two firewalls acting in concert to completely protect a network
Most modern firewall products allow the use of demilitarized zones (DMZ), which provide the functionality of having two firewalls by having different security policies for each attached interface in the firewall. With three interfaces—external network, internal network, and public server network—you can customize your security policy to block connection attempts to your internal network but pass certain protocols to your public servers. This allows you the functionality of two firewalls using a single product. This is sometimes referred to as a *trihomed firewall*. Figure 1.7 shows a trihomed firewall with different security settings for each network.

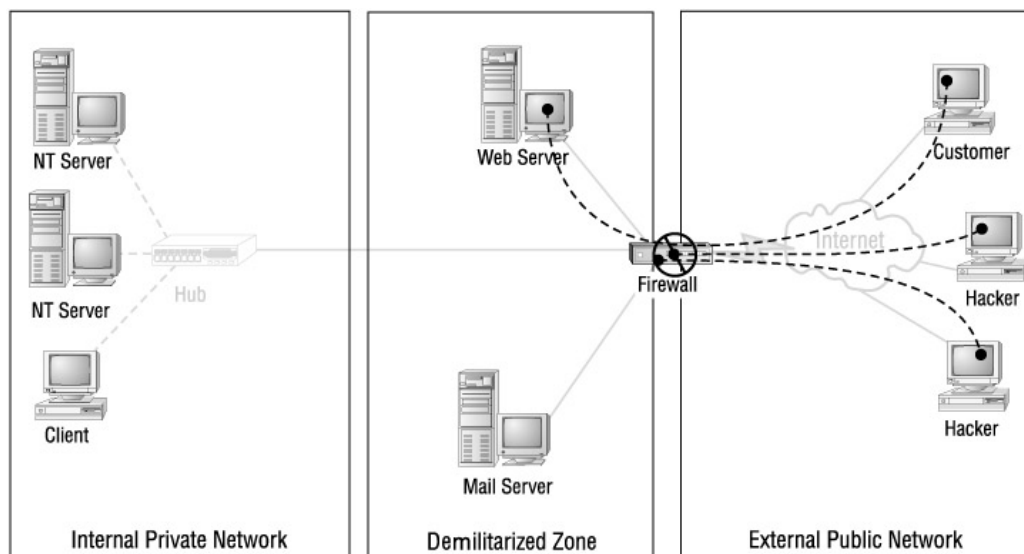


Figure 1.7: A DMZ firewall provides different security for different needs.

Note Always use a DMZ firewall or dual firewalls if you need to provide public services *and* protect an interior network. Every different security policy requires its own firewall or network interface.

Enterprise Firewalls

Enterprise firewalls are those products that share a single, centralized firewall policy among multiple firewalls. Enterprise firewalls allow you to retain central control of security policy without having to worry about whether or not the policy is correctly implemented on each of the firewalls in your organization. The firewall policy is usually defined on a security workstation, and then replicated to each firewall in your organization using some means of secure authentication. Figure 1.8 shows an enterprise with multiple firewalls, one at each Internet connection.

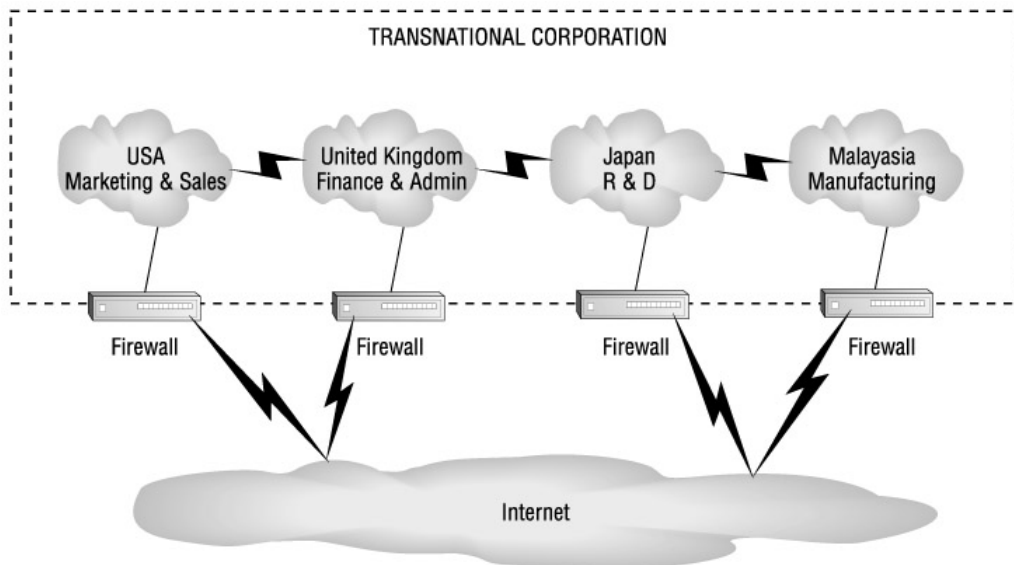


Figure 1.8: Multiple firewalls in an enterprise
Disconnection

The most secure way to provide service on the Internet and access for internal users is not to connect your internal network to the Internet at all, but to have a separate network used only for Internet-related services. Figure 1.9 shows an internal network that is disconnected from the Internet.

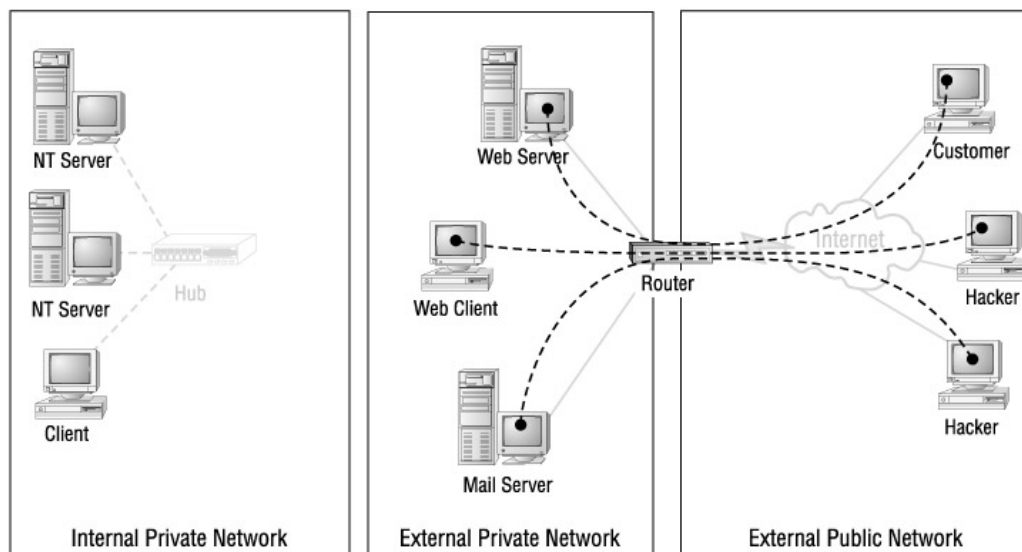


Figure 1.9: The disconnected security model provides the most protection from Internet intrusion. This method is absolutely impenetrable from the Internet because no connection exists between the internal and the external networks. The public-access servers for web, FTP, and mail are located on a small network segment that is attached to the Internet along with a few clients. The client stations contain e-mail, news, and web browsers but no sensitive information. Employees travel to the external clients to check their e-mail, browse the web, or perform any other Internet-related task.

This model has three very important benefits:

- The private network is absolutely secure. Data can't flow freely between the external and internal networks. You may consider putting a high-capacity removable media drive on one

of the clients to facilitate large file transfers when necessary—but this can be a security problem!

- It's free. It doesn't require esoteric software or sophisticated hardware, and you can use outdated computers for the client stations.
- It provides a natural disincentive for employees to waste time surfing the web randomly or downloading content that could cause legal liability problems.

And of course, there is one very important detractor: Employees hate it. They have to travel to access stations, which are typically located in one central area. Transferring files becomes problematic. It can cause a work bottleneck if there aren't enough access stations. Many employees simply won't use it, which reduces the efficiency of e-mail and other such important business tools.

In a nutshell, disconnection is the most secure and the least efficient way to connect your employees to the Internet.

Warning The disconnected security model provides the greatest incentive for employees to blow off your security policy and dial up the Internet with their modem. Make sure your security policy prevents that and that your users understand why you've chosen this model.

Don't attach your network to public networks if it can possibly be avoided. Use the disconnected network model to provide Internet access to your users rather than to your network. Use a web and FTP hosting service rather than computers on your own network to provide your customers with information about your company. This puts the web hosting agency at risk rather than your own network and allows you to provide no public services.

Case Study: Firewall Options

Recently, we were hired to perform ethical hacking against the network of a "famous name" company who used another large multinational to provide third-party security services. The security service involved placing a strong Unix-based firewall at the client's site and performing remote administration and monitoring of the firewall and of hacking attempts

When we began our attack, we used the traditional method of port scanning to determine what we could see in the client's network. Port scanning is easy to detect and was supposed to be part of the monitored services the client received from their security vendor. The scan revealed a potential vulnerability (port 139 was open to one of the internal servers because of "policy drift"—more on that later). We used another traditional method to try to exploit the vulnerability, automated password guessing over the Internet using a common password list. This technique is also very easy to detect and was specifically listed as one of the hacking techniques the service provider monitored and provided protection against.

The password list we used was specially created by hackers from an analysis of hundreds of thousands of exploited user accounts. The hackers created a statistical ranking of the commonality of passwords and created this list that order.

Using this list, our automated password scanner guessed the local administrative password so quickly that we were still in the process of explaining to the client that we would probably not be able to exploit the machine using this technique unless they had really naïve passwords. In this specific case, it hit using the eleventh most commonly used password.

After we screenshot the contents of their web server's hard disk and completed our report, our client

waited for notification from their monitored security service. Notification never came. In fact, our client finally gave up waiting for a call after two weeks and fired the service provider.

In a separate incident, another customer of ours relied on a filtered packet service from their Internet service provider for security. Since the client runs a very small startup business and is strapped for cash, we didn't put up too much resistance to this initially.

As part of our services for them, we made periodic light hacking attempts against their server to make sure no easily exploitable methods could be used to gain access. After having verified the service a number of times, one scan showed that the service had suddenly failed, exposing the NetBIOS session ports of their NT server to the Internet. We mapped a drive connection right to their server over the Internet!

A panicked call to their ISP verified that for some reason the filter had been turned off. The ISP could not explain why or how this had happened and did not know how long the filter had been down. They simply turned the filter service back on and apologized.

Our client decided that they needed to administer security themselves since the ISP could not be trusted to maintain their filtering.

To keep costs as low as possible, we suggested using a Linux-based firewall product or perhaps Linux alone. My client was not comfortable with the user interface however, and decided to go with a Windows NT-based firewall solution. We acquired a machine running Windows NT Workstation and installed Checkpoint Firewall-1. Although Firewall-1 is a more expensive solution, its interface is fairly intuitive. We was able to train the client to administer policy without the help of a consultant, which serves to lower the total cost of ownership. They now have a reliable and secure connection to the Internet.

Chapter 2: Hackers

Overview

Hackers are the reason you need a firewall. An in-depth defense against any adversary requires an in-depth understanding of that adversary, so this chapter will attempt to describe hackers, their motivations, and their methods.

We are hackers. The term "hacker" originally meant someone who understood computers deeply; however, as computers became popular, the media used hacker to refer to those who committed computer crimes, and so the population at large learned the term in the context of the computer criminal. This bothered us ethical hackers, so we began calling malicious hackers "crackers" in order to differentiate them from us. So far, it hasn't worked very well—most people outside the computer security world don't understand the difference.

After much contemplation, we have decided to use the term hackers to refer to anyone who would break into your computer systems because we're not differentiating their motivations. It doesn't matter to us whether the hacker is malicious, joyriding, a law enforcement agent, one of your own employees, an ethical hacker you've paid to attempt to break into your network, or even one of your humble authors. This book is about keeping everyone out. We use the term hacker because it encompasses all these motivations, not just those of the malicious cracker.

Hacker Species

Learning to hack takes an enormous amount of time, as do acts of hacking. Because of the time hacking takes, there are only two serious types of hackers: the underemployed, and those hackers being paid by someone to hack. The word "hacker" conjures up images of skinny teenage boys aglow in the phosphorescence of their monitors. Indeed, this group makes up the largest portion of the teeming millions of hackers. These hackers are now referred to as "script kiddies" in the hacking world, because they download hacking programs called scripts from hacking-interest websites and then try them out in droves against public servers on the Internet. While script kiddies don't do anything innovative, their sheer numbers ensure that any exploits you are vulnerable to will actually be run against you. Because of script kiddies, you simply cannot presume that you won't be found because you aren't famous or in the public eye.

Quite specifically, hackers fall into these categories, in order of increasing threat:

- Security Experts
- Script Kiddies
- Underemployed Adults
- Ideological Hackers
- Criminal Hackers
- Corporate Spies
- Disgruntled Employees

Security Experts

Most security experts (ourselves included) are capable of hacking, but decline from doing so for moral or economic reasons. Computer security experts have found that there's more money in preventing hacking than in perpetrating it, so they spend their time keeping up with the hacking

community and current techniques in order to become more effective in the fight against it. A number of larger Internet service companies employ ethical hackers to test their security systems and those of their large customers. Hundreds of former hackers now consult independently as security experts to medium-sized businesses. These experts are often the first to find new hacking exploits, and they often write software to test or exacerbate a condition. However, unethical hackers can exploit this software just as they can exploit any other software.

We've placed security experts as the lowest threat because if they became a threat, they would, by definition, immediately become criminal hackers. The problem with security experts is the same as with any trusted and powerful (in this specific context) individual—what do you do when they turn on you? In those rare cases where a security expert goes to the dark side, the damage is far reaching and can be so vast that it's difficult to determine exactly what happened. The rarity of this event, not the possible consequences, is what makes security experts a low threat. Even a security expert who is exceptionally ethical can be pissed off; I myself perform self-defense hacking against those who show up with blatant hacking attempts in my firm's firewall logs (which is technically illegal).

Reality Check: Ethical Hackers

In rare cases, the dividing line between a hacker and a security expert is so blurred that they can only be distinguished by their activities. This is the case with groups like the now-defunct L0pht, a cadre of expert hackers that converted into security experts operating a for-profit business. They have, to all appearances, ceased illegal activities, but they write software that is useful both for security administration and hacking; their sympathies lie firmly with the hacking community.

These security experts understand more about hacking than any academic study could ever provide. Their ethos is that the only secure environment is one well tested for security failure. They come under constant fire from those who don't understand that the people who find a problem and publicize it aren't encouraging hacking—they're preventing it.

The work of security experts and hackers in general has had the effect of boosting the Internet's immunity to attack. Imagine what would happen if nobody hacked: Firewalls would be unnecessary, encryption would be unnecessary, and the Internet would be a simpler place. The first criminal hacker to come along would have free and unencumbered access to everything.

The motivation of security vendors, however, can be extremely murky. For example, E-eye is in the business of finding security holes in IIS because they sell software that filters connections on IIS servers. Whenever their research uncovers an exploit that IIS is vulnerable to (and oddly, that their software protects against) they immediately publish the details, knowing full well that a hacker will write an exploit for it, that script-kiddies will download it, that thousands of web servers will be compromised, and that the administrators of those web servers will buy their software. This would be as if the virus scanner companies wrote the very viruses they are supposed to protect your computer against.

Script Kiddies

Script kiddies are students who hack and are currently enrolled in some scholastic endeavor—junior high, high school, or college. Their parents support them, and if they have a job it's only part-time. They are usually enrolled in whatever computer-related courses are available, if only to have access to the computer lab. These hackers may use their own computers, or (especially at colleges) they may use the greater resources of the school to perpetrate their hacks.

Script kiddies are joyriding through cyberspace looking for targets of opportunity and are concerned mostly with impressing their peers and not getting caught. They usually are not motivated to harm you, and in most instances, you'll never know they were there unless you have some alarm software or a firewall that logs attacks. These hackers constitute about 90% of the total hacking activity on the Internet.

If you considered the hacking community as an economic endeavor, these hackers are the consumers. They use the tools produced by others, stand in awe of the hacking feats of others, and generally produce a fan base to which more serious student hackers and underemployed adult hackers play. Any serious attempt at security will keep these hackers at bay.

Script kiddies hack primarily to get free stuff: software and music, mostly. They pirate software amongst themselves, make MP3 compressed audio tracks from CDs of their favorite music, and trade the serial numbers needed to unlock the full functionality of demo software that can be downloaded from the Internet.

Reality Check: Hacker Terminology

If you want to find hackers on the Internet, you need to know the unique words to search for their community web pages. Hackers have adopted the convention of replacing the plural "s" with a "z," specifically for the purpose of making it easy to use a search engine to find their sites. They also use jargon to refer to the various commodities of their trade:

- - warez** Software packages
 - mp3z** Music, from the MPEG-3 encoding scheme used for compression
 - serialz** Serial numbers and unlock codes
 - hackz** Hacking techniques
 - crackz** Patches that will remove the license checks from software packages

Do a web search using these terms to see what you come up with.

Underemployed Adult Hackers

Underemployed adults are former script kiddies who have either dropped out of school or who have failed to achieve full-time employment and family commitments for some other reason. They usually hold "pay the rent" jobs. Their first love is probably hacking, and they are quite good at it. Many of the tools script kiddies use are created by these adult hackers.

Adult hackers are not outright criminals in that they do not intend to harm others. However, the majority of them are software and content pirates, and they often create the "crackz" applied by other hackers to unlock commercial software. This group also writes the majority of the software viruses.

Adult hackers hack for notoriety in the hacking community—they want to impress their peers with exploits and information they've obtained, and to make a statement of defiance against the government or big business. These hackers hack for the technical challenge. This group constitutes only about a tenth of the hacking community, but they are the source for the vast majority of the software written specifically for hackers.

A new and important segment of underemployed adults has recently emerged from the former Warsaw Pact nations. Because of the high quality of education in those countries and the current economic conditions, hundreds of thousands of bright and otherwise professional people hack. Sometimes they have an axe to grind, but most often they are simply looking for something that will make or save them money, like pirated software. Professors, computer scientists, and engineers from those countries have turned their hopes to the Internet looking for employment or whatever else they can find. Students graduate from college, but for lack of employment never graduate from hacking. For similar economic reasons, and because of technological penetration into their society, Israel, India, and Pakistan have recently become hotbeds of hacking activity.

The global nature of the Internet means that literally anyone anywhere has access to your Internet connection machines. In the old days, it took at least money or talent to reach out and hack someone. These days, there's no difference between hacking a computer in your neighborhood and one on the other side of the world. The problem is that in many countries, hacking is not a crime because intellectual property is not strongly protected by law. If you're being hacked from outside your country, you won't be able to bring the perpetrator to justice even if you found out who it was, unless they also committed some major crime, like grand theft of something other than intellectual property.

Ideological Hackers

Ideological hackers are those who hack to further some political purpose. We've added this category since the first edition of this book because in the last three years ideological hacking has gone from just a few verified cases to a full-blown information war. Ideological hacking is most common in hot political arenas like environmentalism and nationalism.

These hackers take up the standard of their cause and (usually) deface websites or perpetrate denial-of-service attacks against their ideological enemies. They're usually looking for mass media coverage of their exploits, and because they nearly always come from foreign countries and often have the implicit support of their home government, they are impervious to prosecution and local law.

While they almost never direct their attacks against specific targets that aren't their enemies, innocent bystanders frequently get caught in the crossfire. Examples of ideological hacking are newspaper and government sites defaced by Palestinian and Israeli hackers both promulgating their specific agendas to the world, or the hundreds of thousands of IIS web servers exploited by the recent "Code Red" worm originating in China, which defaced websites with a message denigrating the U.S. Government. This sort of hacking comes in waves whenever major events occur in political arenas. While it's merely a nuisance at this time, in the future these sorts of attacks will consume so much bandwidth that they will cause chaotic "weather-like" packet storms.

Criminal Hackers

Criminal hackers hack for revenge or to perpetrate theft. This category doesn't bespeak a level of skill so much as an ethical standard (or lack thereof). Criminal hackers are the ones you hear about in the paper—those who have compromised Internet servers to steal credit card numbers, performed wire transfers from banks, or hacked an Internet banking mechanism to steal money.

These hackers are as socially deformed as any real criminal; they are out to get what they can from whomever they can regardless of the cost to the victim. Criminal hackers are exceedingly rare because the intelligence required to hack usually also provides ample opportunity for the individual to find some socially acceptable means of support.

Corporate Spies

Actual corporate spies are also rare because it's extremely costly and legally very risky to employ these tactics against competing companies. Who *does* have the time, money, and interest to use these tactics? Believe it or not, these attacks are usually engaged against high-technology businesses by foreign governments. Many high-technology businesses are young and naive about security, making them ripe for the picking by the experienced intelligence agencies of foreign governments. These agencies already have budgets for spying, and taking on a few medium-sized businesses to extract technology that would give their own corporations an edge is commonplace.

Nearly all high-level military spy cases involve individuals who have incredible access to information, but as public servants don't make much money. This is a recipe for disaster. Low pay and wide access is probably the worst security breach you could have if you think your competition might actually take active measures to acquire information about your systems.

For some, loyalty is bought, and it goes to the highest bidder. Would someone at your company who makes ten dollars an hour think twice about selling their account name and password for a hundred thousand dollars? Money is a powerful motivator, especially to those with crushing debt problems. Many spies are also recruited from the ranks of the socially inept using love, sex, or the promise thereof. Think about the people who work with you—would every one of them be immune to the charms of someone who wanted access?

Remember that these sorts of attacks are not generally perpetrated by your domestic competition, but by the governments of foreign competitors. Domestic competitors prefer the time-honored (and legal) method of simply hiring away those individuals in your company who created the information that your network stores. There's very little that can be done about this sort of security breach, unless you already have employment agreements in place that stipulate non-competition when employees leave the company.

Reality Check: I Spy?

A client of mine recently called me in a panic about a website with a name so similar to his company's own that their customers often accidentally reached it instead of them, which made their company look bad because of its obscene content. When he asked me what could be done about it, I told him that we didn't control the site or the domain name, so there wasn't really anything that we could do about it.

Then he asked me what a hacker could do about it. In abstract terms, I explained the sorts of things a hacker could do in general to take down a website.

Then he asked me if I had the skills to perpetrate that sort of an attack. I explained that while I did, it would be illegal for me to do so, and that my firm didn't sell that sort of expertise.

Then he asked me how much it would cost to convince us to take on that sort of work.

To make a long conversation short, it took me a long time to convince my client that neither my firm nor I would engage in that sort of activity at any price. The incident made me wonder how often hacking attempts are commercially motivated, however.

In an unrelated coincidence, the offending website went down the next day.

Disgruntled Employees

Disgruntled employees are the most dangerous security problem of all. An employee with an axe to grind has both the means and the motive to do serious damage to your network. These sorts of attacks are difficult to detect before they happen, but some sort of behavioral warning generally precipitates them.

Overreacting to an employee who is simply blowing off steam by denigrating management or coworkers is a good way to create a disgruntled employee, however. So be cautious about the measures you take to prevent damage from a disgruntled employee.

Also remember that outsourced network service companies may have policies that make them hard to replace if you decide you no longer wish to retain their services, and that disgruntled small companies tend to behave a lot like disgruntled employees. There's very little that can be done about attacks that come from people with an intimate knowledge of your network, so you should either choose your service providers wisely and exercise a lot of oversight, or require the escort of a trusted employee at all times.

Unfortunately, there's very little you can do about a disgruntled employee's ability to damage your network. Attacks range from the complex (a network administrator who spends time reading other people's e-mail) to the simple (a frustrated clerk who takes a fire-axe to your database server).

Yes, all major operating systems have built-in internal security features that are useful for keeping users in line, but anyone who's ever been an administrator on your network knows all the holes, all the back doors, other people's passwords, and the "administrative" tools that can be used to cause all sorts of local exploits on machines. No version of any major operating system has been immune to "root level" access exploits within the last 12 months, not even the super-hardened OpenBSD. If someone with console access to a running server wants to take it down, it's going down no matter what security measures you have in place.

Accountability and the Law are your friends in this situation. Unlike hackers, it's very easy to track down disgruntled users and apply the force of the law against them. Accountability keeps these attacks relatively rare.

Vectors of Attack

There are only four ways for a hacker to access your network:

- By using a computer on your network directly
- By dialing in via a RAS or remote control server
- By connecting over the Internet
- By connecting to your network directly (usually via a wireless LAN).

There are no other possible vectors. This small number of possible vectors defines the boundaries of the security problem quite well, and as the following sections show, make it possible to contain them even further.

Physical Intrusion

Hackers are notoriously nonchalant and have, on numerous occasions, simply walked into a business, sat down at a local terminal or network client, and began setting the stage for further remote penetration.

In large companies, there's no way to know everyone by sight, so an unfamiliar worker in the IS department isn't uncommon or suspicious at all. In companies that don't have ID badges or security guards, there isn't anybody to check credentials, so penetration is relatively easy. And even in small companies, it's easy to put on a pair of coveralls and pretend to be with a telephone or network wiring company, or even pose as the spouse of a fictitious employee. With a simple excuse like telephone problems in the area, access to the server room is granted (oddly, these are nearly always co-located with telephone equipment). If left unattended, a hacker can simply create a new administrative user account. A small external modem can be attached and configured to answer in less than a minute, often without rebooting your server.

Other possible but rarer possibilities include intruding over a wireless link or tapping some wide area network to which your network is directly attached, like an X.25 link or a frame relay connection.

Solving the direct intrusion problem is easy: Employ strong physical security at your premises and treat any cable or connection that leaves the building as a public medium. This means you should put firewalls between your WAN links and your internal network, or behind wireless links. By employing your firewalls to monitor any connections that leave the building, you are able to eliminate direct intrusion as a vector.

The final direct intrusion problem is that of a hacker who works for your company. This problem is far more difficult to solve than border security, because the perpetrator has a valid account on your network and knowledge of the information it contains. Solving the disgruntled employee/spy problem requires such stringent security measures that your network may become difficult to use for legitimate employees. Many companies find that it's simply not worth the bother and allow the threat to go unchecked.

There is a better way to deal with this remote possibility: strong auditing. Unlike permission-based restriction to resources, an audit approach allows wide access to information on the network and also tracks everything employees do with that access. This doesn't prevent theft or loss of information, but it does show exactly how it occurred and from which account the attack was perpetrated. Because you know the perpetrator directly, you will be able to bring criminal charges against them.

It's most effective to let all employees know that the IT department audits everything that comes and goes in the network for the purpose of security. This prevents problems from starting, since potential miscreants become aware that hacking attempts would be a dead giveaway.

Dial-up

Dial-up hacking via modems used to be the only sort of hacking that existed, but it has quickly fallen to second place after Internet intrusions. Hacking over the Internet is simply easier and more interesting for hackers.

This doesn't mean that the dial-up vector has gone away; hackers with a specific target will employ any available means to gain access.

Although the dial-up problem usually means exploiting a modem attached to a RAS server, it also includes the possibility of dialing into an individual computer with a modem set to answer for the purpose of allowing remote access or remote control for the client. Many organizations allow employees to remotely access their computers from home using this method.

Containing the dial-up problem is conceptually easy: Put your RAS servers outside your firewall, and force legitimate users to authenticate with your firewall to gain access to resources inside. Allow no device to answer a telephone line inside your firewall. This eliminates dial-up as a vector by forcing it to work like any other Internet connection.

Internet

Internet intrusion is the most available, most easily exploited, and most problematic vector of intrusion into your network. This vector is the primary topic of this book. If you follow the advice in this section, the Internet will be the only true vector into your network.

You already know that the Internet vector is solved using firewalls. There's no point in belaboring the topic here since the remainder of this book is about solving the Internet intrusion vector.

Direct Connection

Directly connecting to your network was an esoteric exploit that we didn't bother to mention in the first edition of this book, because someone would have had to somehow sneak an Ethernet cable into your building in order to effect such an intrusion. But recently an amazing new hack-enabling technology has sprung up and become very popular—wire-less networking.

Wireless, especially the extremely popular 802.11b protocol that operates at 11Mbps and is nearly as cheap as standard Ethernet adapters and hubs, has taken root in the corporate world and grown like a weed. Based on the earlier and much less popular 802.11 standard, 802.11b allows administrators to attach wireless access points (WAPs) to their network and allow roaming wireless users (usually attached to laptops) to roam the premises without restriction. In another mode, two WAPs can be pointed at one another to form a wireless bridge between buildings, which can save companies tens of thousands of dollars in construction or circuit costs.

802.11b came with much touted built-in encryption scheme called the Wired-Equivalent Privacy (WEP) that promised to allow secure networking without compromising security. It sounded great. Too bad it took less than 11 hours for researchers to hack. Nobody paid attention at first, so these same researchers released software that automatically hacked it. WEP is so thoroughly compromised at this point that it should be treated as a non-secure connection from the Internet. All wireless devices should be placed on the public side of your Internet, and users should have to authenticate with your firewall.

This leaves just one remaining problem: Theft of service. You can take a laptop down the sidewalks of San Francisco at this very moment and authenticate with any one of over 800 (by a recent count published on Slashdot) 802.11b networks. While you might be outside the corporate firewall, if you're just looking to browse the web, you're in luck. It's especially lucky if you're a hacker looking to hide your trail behind someone else's IP address.

In order to prevent hackers from exploiting your wireless infrastructure to steal Internet access, place your wireless devices inside your DMZ. Then use your firewall to prevent all outbound connections except on those specific ports you need to allow for your servers (think: Just SMTP).

Hacking Techniques

Hacking attacks progress in a series of stages, using various tools and techniques. A hacking attack consists of the following stages:

- **Target Selection** A hacker identifies a specific computer to attack. To pass this stage, some vector of attack must be available, so the machine must have either advertised its presence or have been found through some search activity.
- **Target Identification** The hacker determines the characteristics of the target before actually engaging it. They may achieve this through publicly available information published about the target, or by probing the target using non-attack methods to glean information from it.
- **Attack Method Selection** The hacker selects one or more specific attacks to use against the target based on the information gathered in the previous stage.
- **Attack Progression** The hacker proceeds with the actual attack or series of attacks.

The hacker will attempt to find out more about your network through each successive attack, so the stages above actually feed back into the process as more information is gathered from failed attacks. The major techniques used to accomplish the phases of hacking include:

- Eavesdropping and snooping
- Denial-of-service
- Protocol exploitation
- Impersonation
- Man-in-the-middle
- Hijacking

Once you evaluate your network infrastructure and find weaknesses that a hacker can exploit, you can take measures to shore up your network's defenses.

Eavesdropping and Snooping

The first and easiest things a hacker can do to gain information about your network is simply to listen, and then to ask your network computers information about themselves. The hacker may not even contact your computers directly but instead communicate with other computers that provide services your computers rely on (Domain Name Service computers on the Internet, for example). Networked computers will volunteer a remarkable amount of information about themselves and how they are configured, especially if they are left in their default configurations as supplied by operating system vendors.

Hackers will attempt to exploit any data or network service that is exposed to them. Common hacking practices include (but are by no means limited to) the following activities:

- Password capture
- Traffic analysis
- Network address scanning
- Port scanning
- Finger, Whois, NSLookup, and DNS range grabbing
- SNMP data gathering

Password Capture

Most hacking activities place the hacker at some risk of being detected. One activity that does not pose this threat is eavesdropping on the local networking medium for logon information.

Many networking protocols do not encrypt passwords, allowing any computer on the path between the client and the server to "overhear" the username and password. Not all encrypted logon procedures are safe from eavesdropping either, because (if the logon procedure is naive) a hacker can record the username and encrypted password to send to the server later in a "replay attack" or decrypt the password if the encryption algorithm is flawed or weak.

Eavesdropping requires software that will listen to all of the communications that flow over a network medium, such as Ethernet, rather than just listening to communications that are sent specifically to the hacker's computer. An eavesdropping hacker must also have access to a computer that is situated on a network link with network traffic flowing over it (such as a campus Ethernet or a computer in the server room of an Internet service provider). The more data that flows over the link, the more likely the hacker will capture passwords sent in the clear, i.e. in unencrypted form.

Note While the 802.11b wireless networking protocol broadcasts data in an easily compromised form, the protocol eliminates unnecessary broadcasts by transmitting only those packets directed to the MAC address of the specific wireless adapter involved in the communication. This means that the wireless access points that connect wireless devices to the wired network act like switches rather than hubs. For that reason, 802.11b adapters do not work in the full "promiscuous mode" required for true Ethernet "sniffing" or eavesdropping.

Physical location will not restrict the eavesdropping ability of a hacker who has penetrated other computers on the network. The hacker can install software on those computers that will allow them to snoop as well. The hacker may be typing at a computer in New York while a compromised computer in San Francisco records everything that goes over that remote network for the hacker's later perusal. A determined network intruder may even physically intrude on an otherwise secure LAN and connect a snooping device to the network cable. Casual hackers who are more interested in network joyriding or in finding a place to store their pirated software will seldom exhibit this level of effort (or brave this degree of risk), but other network intruders who might target your network for financial gain could easily do so if you don't take precautions.

Network eavesdropping is a technique hackers can use regardless of the technology used to implement the network. An IPX wide area network is just as vulnerable to someone eavesdropping on network connections as the Internet or an intranet that uses TCP/IP is.

Warning In case you think it might be difficult to remotely install a network sniffer on someone else's network, consider that all versions of Windows Server operating systems include the Network Monitor, a very capable network monitor that can be remotely controlled and is rather easily exploited from afar. While it only records data flowing through the local server, data flowing through servers is typically what a hacker would be looking for.

Snooping Windows passwords over the Internet is surprisingly easy. Microsoft has built in a password Challenge/Response authentication mechanism into Internet Explorer to make secure Intranets easy to build. This mechanism allows a web server to challenge a client for that client's password. The client will respond with the account name of the logged-on user and that user's one-way encrypted password. The password can be decrypted by comparing it to a list of

pre-computed decrypted English words or through a brute-force keyspace comparison. At this point, a hacker has your account name and password, but you would have had to go to the hacker's website to compromise it.

That, too, is surprisingly easy to force. Imagine your boss sent you an e-mail with a link to a website embedded in it and a note saying, "Check this out—they may be competition, what do you think?" Would you click the link? Voila—you're compromised. Forging e-mail is so easy, it's child's play. A hacker can make his e-mail look like it's coming from anyone. All it takes to discover the correct names and e-mail addresses are a few business cards.

E-Mail from Heaven

It's particularly easy to forge Internet e-mail. Try this:

Telnet to a mail server by opening a command prompt and typing **telnet mailserver 25**. Use the mail server configured in your e-mail program if you don't know of another one. Best: Telnet directly to the mail server of the recipient, if you can discover it. To discover it, use the NSLookup tool at the command prompt to look up the MX record associated with the recipient e-mail address's e-mail address.

Type the following at the Telnet prompts, pressing return after each listed line. You won't see text until you press return, and when you type the body text you won't see anything until you press a period by itself and hit enter. Replace the text "YOU@YOURSERVER.COM" with your own e-mail address.

```
MAIL FROM: <GOD@HEAVEN.ORG>
RCPT TO: <YOU@YOURSERVER.COM>
DATA
To: <YOU@YOURSERVER.COM>
From: <GOD@HEAVEN.ORG>
Subject: Concerning your recent activities
Date: Mon, 1 Jan 2000 00:00:01
.
In case you were wondering, forging e-mail is unethical.
QUIT
```

Now that you know how easy it is, you should take forged e-mail warnings very seriously.

Network Traffic Analysis

Passwords aren't the only things a determined hacker will listen for while eavesdropping on network traffic. Quite a bit of information about your network can be determined just from the nature of the traffic in and out of your network (or within your network if the hacker has compromised a computer within your security). Some things a hacker will look for include:

- The IP addresses of the source and destination computers of network traffic.
- The locations of gateways and routers.
- The amount of traffic originating from, being sent to, or flowing through computers identified by the hacker.
- Particular kinds of network traffic going to or from a computer that might identify the computer's function (DNS requests to one computer, or FTP responses from another, for example).

- Network service availability broadcasts (such as NetBIOS browse list updates) that (from an external to a private network) indicate a network security hole or that (within a network) indicate targets for further attack.

The application proxy or Network Address Translation features of a firewall are the best tools for keeping traffic analysis from revealing too much about your network. The firewall will make all of the Internet (or other public network) traffic appear to come from one computer. A hacker from outside will not be able to determine the true extent of your network behind the firewall. You must also configure your firewall not to pass service availability broadcasts beyond your network boundary.

Network Address Scanning

Nearly all hacking attacks these days start with network address and port scanning. The hacker will specify a beginning and ending address to scan, and then the hacker's computer program will attempt to establish a connection to a computer on each of those network addresses in turn. If a computer answers from any one of those addresses the hacker has found another target.

All network technologies that specify an address of one kind or another for each computer on the network are vulnerable to this kind of attack. TCP/IP is the network technology most often scanned by hackers, and tools to scan TCP/IP are widely available. Other technologies such as NWLink, X.25, and FDDI are equally susceptible, if the hacker is willing to find or create the tools necessary to perform the scan.

The best way to foil this kind of attack is to watch for it. A network administrator who determines that this kind of attack is in progress can take steps to halt it, including configuring gateways or routers to discard network traffic from the offending host(s).

You need to configure gateways, packet filters, and routers to log connection requests to hosts that do not exist on your network. Periodically examine log data for network address scanning, and (if the logging software supports it) configure a network alert that will signal if a scan is in progress.

Port Scanning

Once a hacker has identified a target computer, the hacker will attempt to determine which operating system it is running and what services it is providing to network clients. On a TCP/IP-based network (such as the Internet), services are provided on numbered connections called *sockets*. The set of sockets to which a computer responds often identifies the operating system and supported services of the target computer.

There are a number of tools available on the Internet that a hacker can use to determine which sockets are responding to network connection requests. These tools try each port in turn and report to the hacker which ports refuse connections and which do not. The hacker can then concentrate on ports corresponding to services that are often left unsecured or that have security problems.

Port scanning can reveal which operating system your computer is running because each OS has a different set of default services. For example, by scanning the TCP ports between 0 and 150, a hacker can discern Windows hosts by the presence of port 139 in the scan list, NT/2000/XP hosts by the presence of port 135 in the list, and various Unix hosts simply by the presence of TCP/IP services like port 23 (Telnet), which Windows computers do not install by default. This information tells the hacker which tools to use to further compromise your network.

The defense for port scanning is the same as for network address scanning—watch for connection

attempts to unsupported ports and then deny access to the computers that are doing the scanning. Periodically examine log data for port scanning, and (if the logging software supports it) configure a network alert that will signal if a scan is in progress. Consider setting up a server whose only purpose is to be exploited (a "honey pot") and then simply log every connection attempt to it. If you don't set a DNS name for it and don't advertise its existence, every connection attempt to it is the result of a hacking attempt.

Finger, Whois, NSLookup, and DNS Zone Transfer

There are a number of network services that hackers will use to gather information, if the ports used by those services are enabled on your Internet host. The Finger and Whois services are hacker favorites because they supply the account name and personal contact information for users of network computers. These are useful services for people who need to contact members of your organization or who need to find an e-mail address for a network user, but hackers will take usernames returned by these services and then attempt to break into those accounts by trying commonly used passwords.

Note By default, Windows does not support Finger or Whois. If you support Unix computers in your network, however, you should either disable these services or curtail the information they return. You can install software for Windows that provides these services, but you probably shouldn't.

Few network users will miss the Finger and Whois services, but the same cannot be said for the DNS service. The DNS service is required by Internet client software to convert human-friendly Internet names such as <http://www.microsoft.com/> into computer-friendly IP addresses such as 10.1.1.2. Without the DNS service, the Internet would be useless for public services.

Windows servers do support the DNS service. Most networks that support the use of Internet tools within the network (instead of just the use of Internet tools to connect to services on the Internet) will include support for DNS. A smaller network can rely on an external DNS server to provide Internet name service translation for its clients, but a large IP network or an IP network behind a firewall is difficult to manage without a DNS server of its own. Active directory requires DNS as well.

Hackers can use a DNS service to discover the structure of your network. Since DNS records the IP addresses and Internet names of all of the servers on your network, a hacker can attain a list of the most important computers in your network. The NSLookup tool is a standard Internet program for interrogating DNS servers, and a hacker can craft a program based on the NSLookup that would even make the hacker's computer appear to be a peer DNS server that needs information. Your task is to configure security in a way that allows clients from within to access the DNS server and get the information they need, but also prevents computers from outside your network from getting that information.

Tip Foil hackers attempting to gather information about the interior of your network by using different public and private DNS servers in your network. Establish internal DNS services for inside clients, and then set your internal DNS servers to forward to an ISP's DNS server for resolution of names not known to your interior DNS machines. Then block inbound DNS requests from the Internet to your inside domain name servers. To publish the names of public servers, either use your ISP's DNS servers or set up an additional pair of outside DNS servers in your DMZ that is used exclusively for public addresses.

The security problem is compounded by the fact that DNS is a hierarchical service. If one DNS server does not have the answer to a query, it will ask the next server up or down the DNS tree.

This means that in a traditionally configured network, a DNS service within your firewall will need to be able to communicate with DNS servers outside the firewall. DNS servers are also configured to transfer blocks of Internet name and address data using a feature called Zone Transfer. In addition, many websites will not respond to Internet requests from client computers that don't have DNS reverse mappings, so the Internet servers that run those sites must be able to connect to your DNS server (via their DNS server or the DNS server up the tree from yours) to verify that the DNS reverse mapping exists.

A firewall can solve these problems by handling name translation inside your network. If your network requirements mandate that computers external to your network must be able to resolve IP addresses for computers inside your firewall or vice versa (if you use a software package that does not support use of a proxy server, for example), you should configure your firewall to disallow connections to your DNS server for all external computers except that of the DNS server up the tree from yours. You should also disable zone transfers for all DNS servers except those within your security domain. Chapter 10, "The Ideal Firewall," covers specific policies you should enable in your firewalls based on your level of risk.

SNMP Data Gathering

The Simple Network Management Protocol (SNMP) is an essential tool for managing large TCP/IP networks. SNMP allows the administrator to remotely query the status of and control the operation of network devices that support SNMP. Unfortunately, hackers can also use SNMP to gather data about a network or (as described in the next section) interfere with the operation of the network.

Again, a firewall solves the problem. There's little reason why any computer outside your network should need to query SNMP, so simply block SNMP messages through your firewalls. Chapter 10 will show you exactly which services you should block with your firewall.

Denial of Service

The next easiest attack on your network is to disable some aspect of it or even bring the entire network down. The hacker may be merely interested in inconveniencing your organization, or they may have a more sinister purpose. In any case, you should remember that it is much easier for one computer to impersonate another computer (see the next section on impersonation) if that other computer is disabled.

There are a number of methods a hacker can use to disable a computer or a service provided by a computer. Most of these methods affect computers using TCP/IP, because TCP/IP is the most widely used internetwork protocol and because the most pressing hacker threat is from the Internet. Methods hackers can use to disable computers or computer services include these:

- Ping of Death (malformed ICMP packets)
- SYN (Synchronize Connection Establishments) Attacks and ICMP (Internet Control Message Protocol) flooding
- Service Specific Attacks
- DNS Redirection
- Route redirection: RIP (Router Information Protocol), BGP (Border Gateway Protocol), and ICMP
- SNMP reconfiguration

Ping of Death

Perhaps the most ominous sounding of Network layer attacks is the aptly named Ping of Death. A specially constructed ICMP packet that violates the construction rules can cause the recipient computer to crash if that computer's networking software does not check for invalid ICMP packets.

The only solution for computers outside your gateway (or the gateway computer itself) to resist the Ping of Death is to use a version of the operating system that is not susceptible to the Ping of Death. You can shield computers inside your network by not passing ICMP echo packets through your firewall.

SYN Attacks and ICMP Flooding

Another way hackers disable the networking capability of computers is by overloading the network protocol software of the target computer with connection attempts or information requests. The initial IP packet of a TCP connection attempt is simple and easy to generate (a distinguishing characteristic of these packets is that they have the SYN bit set). Responding to a connection attempt takes more compute time and memory space than generating the packet does, because the receiving computer must record information about the new connection and allocate memory for connection data. An attacker can send one SYN packet after another to a target computer, and that target computer will then be unable to process other connection attempts from legitimate users because all of its available time and memory will be spent processing SYN requests.

A similar network protocol attack is ICMP flooding, in which the hacker sends a constant stream of ICMP echo requests to the target computer. The target computer then spends most of its time responding to the echo requests instead of processing legitimate network traffic.

Keep your firewall and operating system software updated to prevent against these attacks. You should configure your firewalls or servers to log instances of extremely frequent SYN connection attempts or an abnormally high volume of ICMP traffic in order to protect operating systems outside your firewall that may be vulnerable to these attacks.

Service Specific Denial of Service Attacks

Hackers are usually not interested in crashing your computer. The hacker may instead be more interested in shutting down one of the services supported by your network-connected computer (perhaps in order to impersonate that service, as described in the next section on impersonation).

Although any service provided by your computer may be the target of a service-specific attack, there are four services that hackers are particularly attracted to, because they are either fundamental components of a TCP/IP network or fundamental components of Windows networking. The four services are RPC, NetBIOS, DNS, and WINS. Other services, such as Chargen or Time, do not provide a sufficiently rich environment for a hacker to have any real chance of using the service to break into or take down your computer.

Network clients connect to specific ports for each network service, and each service expects the network client to send the data to the service in a specific format. The DNS service, for example, expects that data sent to the DNS port from the client is formatted in a different manner than it is for WINS requests, and DNS will not be able to respond properly to WINS requests sent to it.

This is much like real-world services such as those provided by the Department of Motor Vehicles and the Social Security Administration, each of which needs different information from you in order

to perform their services, and each of which has different forms for you to fill out. You could send a form requesting a duplicate Social Security card to the DMV, but you would neither get a Social Security card nor a driver's license in return. You must send the right form to the right service.

While the repercussions of sending misleading or incorrect information to government institutions can be severe for the perpetrator, it will have negligible effects on the operation of the government service. However, sending incorrect or nonsense messages to a network service can crash the service, and it is difficult to track back to the hacker, especially if the hacker is using obfuscatory techniques, such as source routing, or has suborned another computer into being a relay for hacker activity.

Many implementations of DNS, RPC, and WINS are particularly vulnerable to receiving random information at their ports. Some implementations of DNS also crash if they receive a DNS response without having first sent a DNS request. You can protect against unsolicited DNS responses by only allowing authorized external hosts to communicate with your DNS server.

The NetBIOS service of Windows is vulnerable to an out-of-band attack sent to the Net-BIOS ports. NetBIOS ports should not be accessible to computers outside your network at all, so the best solution to this problem (after installing the latest version of the operating system software) is not to bind NetBIOS to network adapters that can be reached from outside your network.

DNS Cache Pollution

An additional DNS service attack that deserves special mention is DNS cache pollution. A hacker can observe a computer that provides DNS services (using techniques described in the previous section on eavesdropping and snooping) and determine the sequence used by the computer to provide query IDs for recursive DNS queries. The hacker can then forge a response to the next DNS query that contains invalid information or information that will redirect Internet traffic to a computer the hacker has already suborned. (The hacker may have to perform a denial-of-service attack on the DNS server being queried in order for the substitution to be accepted by the querying, targeted DNS server.)

This sort of attack can cause client computers that rely on the DNS server to not be able to resolve Internet names into valid IP addresses. That alone can cause problems on a TCP/IP network. More dangerous, however, is when a hacker populates the DNS server with valid IP addresses that are different from the correct IP addresses, especially if the hacker controls the computers at those addresses. A DNS cache pollution attack can therefore be the beginning of an impersonation attack on computers in your network.

Route Redirection (RIP, BGP, ICMP)

A hacker can cause a great deal of havoc in your network if the hacker can get control of your network's routers. Routers direct the flow of information within your network (as well as in and out of it), from information stored in their routing tables. By making changes to those routing tables, a hacker can isolate parts of your network and direct network traffic out of your network.

Routers must adapt to network conditions in order to maintain network functionality in the face of slowdowns or failures in network links. The routers in your network will exchange information about routing conditions, accept routing updates from network administrative programs, and communicate with routers outside your network if you allow them to. These routing updates are transmitted using a routing protocol, usually RIP, OSPF, or BGP.

RIP has no authentication capability. If a hacker can communicate with a router that uses RIP to update its network information, then the hacker can easily reconfigure the router to deny service to computers in your network or redirect the network traffic from computers in your network. OSPF provides more security than RIP does, and BGP is fairly secure about who it will communicate with in order to update routing tables.

Another way a hacker can get your computers to send data to the wrong address is to send ICMP redirect packets to the computer. An ICMP redirect packet instructs the computer that an IP packet is being sent to the wrong router and that there is another route to the destination address that is either more efficient, faster, or capable of avoiding a network problem. It is difficult to forge ICMP packets, however, because they must appear to come from the router closest to the originating computer.

SNMP Reconfiguration

Many network devices, including Windows Server computers (if you install the SNMP service for them) can be managed remotely using SNMP. In addition to data snooping, a hacker can use SNMP to reconfigure your network to deny service to network computers or even to route data out of your network—depending on the SNMP features of the device the hacker gains control of.

Protocol Exploitation

Protocol Exploitation is currently the most popular form of hacking on the Internet. Protocol exploitation is an attack based on exploiting a bug in a public service in order to gain more access than would normally be allowed.

Buffer Overruns Explained

The most common form of protocol exploitation is the venerable buffer overrun, which is an artifact of the way that modern compilers of certain programming languages create programs.

In a C or C++ program, when a function allocates a local variable—say, to copy data into—that variable is placed on the program's stack (e.g., its temporary data region). Then, when the function calls another function, the subsequent function's return value is placed on the stack behind the local variable of the calling function.

Now, in the case where more data is copied into a variable than the variable was allocated to store, the extra data will overwrite the return value of a subsequently called function, so that when that function terminates and its return pointer is removed from the stack, the return pointer will contain a nonsense value, and the computer will attempt to continue executing at that nonsense location.

Normally, this would simply cause a program to crash, and this sort of problem is the source of many of those "access violations" and "general protection faults" that to which Windows has been historically susceptible.

But in the case where someone who has analyzed the running characteristics of the program in question has specifically crafted the extra data, the overwritten return value can be crafted to jump to a portion of the program that isn't garbage—for example, back to the beginning of the copied in data itself.

This means that a hacker can, in many cases, simply push data that represents executable code to a service (e.g., sending a URL to a web server), push enough extra data to be certain to overwrite a

return pointer, and have that pointer point back to the beginning of the buffer so that the code is executed.

This code is then executed in the user context of the application that copied it in. By targeting services that run under "root" (Unix) or "Local System" (Windows) security contexts, the code is then free to perform nearly any task. Typically, the code would simply execute a call to open a security hole for further exploitation, such as patching the registry to allow the command console to receive commands from the web service, or passing a system call to download a fully exploitable Trojan horse.

How does one "push data into a service?" Simple. When you type a URL into your web browser, that URL is sent to the web server where it's copied into buffers. When you send an e-mail to a mail server, the contents of that e-mail are copied into a buffer. Any communication whatsoever with a public server is an opportunity for hackers to look for a buffer overrun exploit.

Some programming practices are so susceptible to attack that hackers simply scan the binary code of a program for the signature of susceptible practices, such as using the venerable `strcpy` function in the C programming language, to copy data between buffers. This function doesn't perform any boundary checking, it's commonly used in buffer applications, and it's easy to hunt for. So hackers simply search the code of the web server application looking for the telltale signs of this weakness, and then analyze the code at that point with the same software that programmers legitimately use to debug their programs. Once they've found a candidate weakness, they write a piece of exploit code that fits the buffer size in question and overwrites enough data to get to a return pointer, and then test it on their own machines. Once they've got the right sequence, they package it as a script and upload it to sites that serve the teeming millions of script kiddies—who then download it and run it against your server.

If it seems like this would take a lot of effort, consider that just two web servers, Apache and IIS, currently account for 90% of all web servers on the Internet, and a single e-mail service application, Sendmail, performs 80% of the e-mail forwarding on the Internet. Those three applications alone are constantly analyzed by thousands of highly skilled hackers. Given the odds, it's surprising how *infrequent* truly new attacks really are.

So the question arises: Why not just check to make sure that the amount of data copied is within the allowable limits? Answer: Good question.

In reality, most programmers never consider the fact that code other than what they write might be executing inside their program. Why would they? Aside from someone actually maliciously changing their code while it executed, this sort of thing would never occur and therefore usually doesn't need to be checked for, so most programmers who learn to program in the safe context of a University computer lab don't get in the wasteful habit of checking everything. Many programmers also presume that the language or the compiler performs these checks for them, which is true for many languages, but it's not so in the case of C and C++, the languages used to write the vast majority of Internet services.

Weinberg's second law states: "If builders built buildings the way that programmers write programs, the first woodpecker to come along would destroy civilization." Well, the woodpeckers have arrived.

What can you do about it? Use an uncommon web server. Use an uncommon e-mail server. Use proxy servers or security software designed to filter these sorts of attacks. And don't host public services inside your private network.

We have one client who uses IBM's Lotus Notes for e-mail and Domino for their web services. As yet, they've never been hacked (whereas many of our IIS clients have), and we've never heard of any serious exploits against Lotus. It's not that there's anything fundamentally more secure about Notes or Domino, but their relative obscurity keeps them out of the hacking limelight. From a security standpoint, it's security through obscurity, but in this case it dramatically reduces your risk of intrusion.

Impersonation

Impersonation is the next step for a hacker to take if the hacker still doesn't have access to your network computers. The goal of a hacker is to penetrate your network security and get at the information or resources on the computers in your network. These attacks are a lot harder and a lot rarer than protocol exploitation attacks. These attacks are used when a specific target is the goal, rather than when the desired outcome is the random mayhem caused by protocol exploits.

Merely snooping on your network traffic may give the hacker enough information to log on to your network. If that does not work, the hacker may reduce the functionality of your network via a denial-of-service attack, causing computers on your network to reveal enough information to allow the hacker to break in. The hacker might also pursue a denial-of-service attack just to inconvenience users of your network. Ultimately, you may not be susceptible to any well-known protocol exploitation attacks. By impersonating another computer that the computers on your network trust, the hacker's computer may be able to trick your computers into revealing enough information for the hacker to get through your network security. Alternatively, by impersonating another computer, the hacker's computer may be able to trick one of your computers into executing a command that weakens your security enough to let the hacker in. The tactics a hacker may use depend on the computer or service that the hacker is attempting to impersonate, which include the following:

- Source routed attacks
- DHCP, WINS, and DNS service impersonation
- Password playback, server impersonation, and password capture

Source Routed Attacks

The TCP/IP protocol suite includes a little-used option for specifying the exact route a packet should take as it crosses a TCP/IP-based network (such as the Internet). This option is called source routing, and it allows a hacker to send data from one computer and make it look like it came from another (usually more trusted) computer. Source routing is a useful tool for diagnosing network failures and circumventing network problems, but hackers too easily exploit it and so you should not use it in your TCP/IP network. Configure your firewalls to drop all source-routed TCP/IP packets from the Internet.

The hacker can use source routing to impersonate an already connected user and inject additional information into an otherwise benign communication between a server and the authorized client computer. For example, a hacker might detect that an administrator has telnetted onto a server from a client computer. If that administrator is at a command prompt, the hacker could inject a packet into the communications stream would appear to come from the administrator and would tell the server to execute the change password command—thereby locking the administrator account and letting

the hacker in.

The hacker also might use source routing to impersonate a trusted external DNS server and send DNS updates to your DNS server. This redirects all of the network clients that rely on the DNS server to translate Internet names into IP addresses, so that the client computers go instead to a hostile server under the control of the hacker. The hacker could then use the hostile server to capture passwords (as described later in the section on password playback, server impersonation, and password capture).

DHCP, WINS, and DNS Service Impersonation

Another tactic a hacker can use to penetrate your network is to impersonate a service that your client computers get configuration information from at boot time. Network clients can be set up to get their configuration (including the location of the default gateway, DNS, and WINS servers) from a DHCP server, so a hacker who can impersonate a DHCP server can redirect your network clients to talk to almost any hostile host. By impersonating a WINS server, the hacker can return invalid or hostile IP addresses for NetBIOS computer names. By impersonating a DNS server, the hacker can return invalid or hostile IP addresses for Internet names as well.

In order for a hacker to impersonate a DHCP, WINS, or DNS server, the hacker must get control of one computer within your network and then initiate a denial-of-service attack against the legitimate DHCP, WINS, or DNS target computer. Once the target computer goes down, the computer controlled by the hacker can begin satisfying DHCP, WINS, or DNS requests in its place. This is just one way that a hacker can use one compromised computer in your network to penetrate your network security further and gain control of other computers in your network.

A DHCP, WINS, or DNS impersonation attack on your network relies on other attack methods to succeed. The hacker must first gather information about your network in order to identify targets, and then cause a denial of service on the service being impersonated. After that succeeds, the hacker must either gain control of at least one computer in your network that will be used to take the place of the server being impersonated, or redirect network traffic to an external computer that can take the place of the server being impersonated. The defensive measures you put in place to stop denial-of-service attacks and to restrict information about your network will help prevent an impersonation attack as well. You should also watch your network traffic for DHCP, WINS, or DNS services being hosted by unauthorized computers in your network, and you should take swift action to shut down any unauthorized servers.

Server Impersonation, Password Capture, and Password Playback

If the hacker has observed an encrypted logon session to one of your computers, they may not know the username and password being used to log on but might be able to fool your system anyway. The hacker might simply record the encrypted logon credentials and send those same credentials to your computer later. It won't matter that the hacker can't discern what the password is, because the receiving computer expects it in encrypted form anyway.

Older networking protocols are vulnerable to this sort of attack. This attack can be defeated by using challenge and response authentication for passwords or by "salting" the password with the current system time so that it can't be decrypted beyond a minute (or whatever the allotted time resolution is) from the original encryption time. Of course, that requires perfectly synchronized clocks, which requires the Network Time Protocol, which is subject to exploitation...

With challenge and response authentication, the password is never transmitted. Rather, the client

indicates that she would like to log on. The server transmits a unique number to the client. Both computers encrypt that number using the client's password as a key. The client transmits the encrypted number back to the server. If the encrypted results match, then the same key was used to perform the encryption and the server knows that the client knows the correct password. By encrypting a random number, the results will be different each time, making it impossible to derive the password used to encrypt it through mechanisms like snooping.

Windows and most modern versions of Unix use this sort of password encryption and authentication by default for NetBIOS connections from network client computers. Unfortunately, Windows also supports an older LAN Manager authentication protocol. Networking clients may elect to use an older protocol if they inform a Windows server that they do not support the Windows NT Challenge/Response protocol. Hackers can exploit Windows support for this weaker protocol against newer computers by forging a response packet that appears to come from the server (using source routing, rerouting, or a man-in-the-middle position) and that instructs the modern client to use the weaker LAN Manager protocol. This way, the hacker can make the client use an easily cracked password encryption method or even instruct the client not to use password encryption at all. The hacker can then eavesdrop on the resulting logon traffic and capture the password used by the client to log on to the server. The best solution to this security problem is to configure your Windows computers not to accept LAN Manager authentication.

Many older UNIX protocols, like Telnet, also don't make use of challenge and response authentication. Hackers can simply sniff these passwords off the network. E-mail clients like POP3 and IMAP, as well as Basic AUTH, transmit the user's password in plain text, so users who check e-mail accounts over the Internet are susceptible to having their passwords sniffed as well. If those passwords are the same as their network accounts, their account information is compromised.

Note Never, ever use the same account names and passwords on your local network as you use for e-mail services. Because most e-mail services transmit passwords in the clear by default, hackers can easily compromise account names and passwords. If those account names and passwords match your internal network passwords...

One limitation of the NT Challenge/Response protocol as it exists now is that a hacker can set up a server for capturing passwords and either entice users to connect to the server (the hacker can set up a website, for example, that accepts Windows NT Challenge/Response authentication), or the hacker can use denial-of-service attacks to redirect network connections to a computer that is masquerading as a valid server for your network. When the hacker configures the deceptive server, the hacker does not have the passwords for the user accounts that will attempt to connect to it. The hacker can specify one number that the server will always send to the client computers as a challenge, however, and can pre-compute passwords encrypted with that seed from dictionary files. If an unsuspecting client computer sends an encrypted password that matches a computed dictionary value the hacker has calculated, then the hacker has found a new username and password to use to get into your network. Currently, the best solution to this kind of network attack is for network users to never select passwords that might show up in a dictionary, that might be easily guessed, or that are under eight characters in length.

Man-in-the-Middle

A special case of the impersonation attack is the man-in-the-middle attack, where the hacker operates between two computers on your network, or between a client computer on the Internet or other WAN network and your server computer in your secure LAN. When the client computer opens a connection to the server computer, the hacker's computer intercepts it (perhaps via a DNS or DHCP impersonation attack or by rerouting the IP traffic from the client to a compromised

computer). The hacker computer opens a connection on behalf of the client computer to the server computer. Ideally (from the hacker's point of view), the client will think he is communicating with the server, the server will think it is communicating with the client, and the hacker computer in the middle will be able to observe and alter all of the communications between them.

Depending on the nature of the communications, the hacker computer may be able to use a man-in-the-middle attack to gain greater access to your network. For example, if the connection is an administrator-level Telnet session into a server computer from a client computer, the hacker computer in the middle could (after passing through the logon credentials to gain entry to the server) download the password file from the server to the hacker computer, instead of uploading HTML pages that the administrator may wish to place on the server computer.

On an insecure network such as the Internet, it is difficult to defend against a man-in-the-middle attack. Fortunately, a successful man-in-the-middle attack is also difficult to construct. The measures you take to protect your network against data gathering, denial-of-service, and impersonation will help protect you from a man-in-the-middle attack. Nevertheless, you should never connect to your network using an administrative account over an insecure network.

You can use encryption to create secure communication links over a TCP/IP network, and you can use third-party authentication packages (such as S/KEY provided by Bellcore or SecureID provided by Security Dynamics) to ensure that your client computers are communicating directly with a trusted host computer (and vice versa).

Hijacking

One last hacker trick is the hijacking of an already established and authenticated networking connection. This can occur at two layers of the networking protocol—at the TCP connection layer and at the SMB or NFS Session layer. In order for a hacker on the Internet to hijack a network share connection, the hacker will have to do both, because SMB uses TCP ports to make the connection.

In order to hijack an existing TCP connection, a hacker must be able to predict TCP sequence numbers, which the two communicating computers use to keep IP packets in order and to ensure that they all arrive at the destination. The hacker must also be able to redirect the TCP/IP connection to the hacker computer, and also launch a denial-of-service attack against the client computer so that the client computer does not indicate to the server that something is wrong. In order to hijack an SMB session (such as a drive mapping to a file share), the hacker must also be able to predict the correct NetBIOS Frame ID, Tree ID, and the correct user ID at the server level of an existing NetBIOS communications link.

While an exploit of this nature is theoretically possible, tools for hijacking SMB connections are not readily available to the garden-variety hacker (as opposed to TCP hijacking tools, which can be downloaded from the Internet). A properly secured Internet site will not expose SMB to the Internet, however.

Case Study: The Metamorphosis

The authors of this book were hackers in high school. We started on simple eight-bit computers like Ataris and Commodores, trading tricks, tips, and software amongst ourselves for such nefarious purposes as copying write-protected floppy disks and unlocking secret codes for games. Back then, the protections afforded to software were so simple that a 15-year-old stood a reasonable chance of cracking them. I remember that we once disassembled and printed out a piece of software in its entirety, and then pored through the code by hand to find the routines that checked for the presence

of a hardware key.

Back then, nobody outside of a few government agencies and universities knew about the Internet, so modems were our gateway to the wide world of computers. Hackers would hang out on dial-up servers called Bulletin Board Systems (BBS), the archaic equivalent of chat servers, to trade software, secrets, and chat amongst themselves. The coolest of these boards were never in your local city, however, so you had to pay long-distance fees to dial into them.

Hacking consisted of "war-dialing" random numbers, looking for the carrier tone of a modem. If a human picked up the phone, the computer would drop the line and proceed to the next number in the list. The first pass consisted of simply browsing for carrier tones and printing the phone numbers that were answered by computers. Once a list of computers was generated, we'd browse that list looking for interesting targets.

Because we had to use direct-dialed telephone lines to connect our computers to BBS hosts, we needed a way to pay long-distance charges without them showing up on our parents' phone bill. This, too, was an easy problem to solve—we simply dialed a long-distance provider and supplied a random number as the access code. When the number worked, we'd print it out. When it didn't work, we just tried again. By leaving a computer running over night while we slept, the method could generate about 10 valid access codes a day. We'd use each access code just once so that by the time anyone knew the call had been illegal, it would be too late to trace.

When the FBI raided the home of a 12-year-old friend of ours because he repeatedly used an illegal phone card number that we generated, we realized the seriousness of what had been just fun for us. His parents were held liable for over 20 thousand dollars in phone charges and fines, and he lost his computer until he reached the age of 18. That (along with a few other hacks that went south) ended our hacking careers before we left high school.

Hacking did prepare us for the rigors of network integration, however. We learned a lot of important information early, so when we ran into similar problems as adults, we already knew how to solve them. When Local Area Networks first came on the scene in the '80s, we already understood network technology—so it was easy for us to move into the field.

Chapter 3: TCP/IP from a Security Viewpoint

Overview

If you are reading this book, you should have a good understanding of how computers work and a working knowledge of how to use Internet tools such as web browsers, Telnet, and e-mail. In addition, you're probably already aware of the need to protect computers on your network from exterior threats, while still allowing your web and e-mail traffic to traverse your connection to the Internet. You may install a firewall to secure your network, but to configure it correctly you must know just how your computer connects to other computers and downloads web pages, exchanges e-mail, or establishes a Telnet session. You'll also need to know how to set firewall rules to differentiate the legitimate network traffic of your network users from the illicit access of hackers and other external threats. As TCP/IP is the mechanism by which your computer communicates with the rest of the Internet, you will need to have more than a passing familiarity with it. This chapter will give you a better idea of what is going on behind the scenes.

You do not need to absorb all of the information in this chapter before you set up your firewall—some of the information here is more detailed than you will need initially—but by reading it, you can get a good idea of what sort of network vulnerabilities you should be concerned about. For example, if you have an 802.11b access point on your LAN, you really should read the Wireless section in the Physical Layer. Also, when new threats arise on the Internet, you will find the information in this chapter an excellent reference for understanding how the threat works, (be it a virus, worm, Trojan horse, or protocol exploit), whether or not your network is at risk, and what to do about it if it is.

This chapter explores the workings of the TCP/IP stack that transports data across the Internet. The next chapter examines the common protocols such as HTTP and SMTP that use TCP/IP. You should be familiar with both the stack itself and the protocols that use it in order to properly set up your firewall.

You Need to Be a TCP/IP Guru

But why do you care how TCP/IP works if you aren't a computer programmer or network engineer? You should care, because the hackers attempting to get past your network security often *are* computer programmers or network engineers (self-taught or otherwise), and in order to stop them you need to understand and correct the weaknesses in TCP/IP or higher-level protocols that they will attempt to exploit. In other words, know what your enemy knows.

You don't have to be intimidated by the network technology; you just need to know enough to keep the hackers out, not so much that you can recreate a network from scratch. If you were planning the defense of a castle, you wouldn't need to know how to build the stone walls or forge the swords, but you would need to know where the openings were, how the invading barbarians typically attacked a castle, and what defenses you had at your disposal.

Similarly, you don't need to drop everything and learn how to write device drivers in C, nor do you need to pore over the Internet RFCs that describe the protocols you use. You should know which protocols your network supports, however, and you should have a basic understanding of how those protocols interact with your firewall, the client computers on your network, and with other computers outside your firewall on the Internet. You should understand the risks (and benefits) of opening ports on your firewall for the various services your network clients would like to use. You

should be aware of the limitations a firewall places on network traffic, and you should understand which protocols hackers easily subvert and which ones they can't.

TCP/IP Rules

What is the big deal about TCP/IP anyway? Why, with its acknowledged weaknesses (we'll get to them in a moment), is the world using TCP/IP to "get wired" instead of another protocol, such as IPX/SPX or SNA? TCP/IP has won out over other protocols that might have competed for world domination for the following reasons:

- **TCP/IP is packet based.** With TCP/IP, many communicating computers can send data over the same network connections. The alternative is to use switched networks, which require a dedicated circuit for every two communicating devices. Packet-based networks are less costly and easier to implement. They typically don't guarantee how much bandwidth the communicating devices will get or what the latency will be. The market has shown, through the Internet, that low cost is more important than guaranteed performance.
- **TCP/IP provides for decentralized control.** Every network that communicates via TCP/IP gets a range of numbers to use for the computers on that network. Those numbers, once assigned to the organization that requested them, are under the control of that organization for assignment, reassignment, and even sub-allocation to other organizations. Internet service providers, for example, get a block of numbers and then dynamically allocate them to callers as they attach to the ISP. Similarly, the Internet domain names, once assigned to an individual or organization by a top-level Internet authority, can be further sub-allocated locally without top-level intervention or authorization. If you own sybex.com, for example, you can assign www.sybex.com to one computer, ftp.sybex.com to another, and mail.sybex.com to a third. Similarly, utah.edu is subdivided by the University of Utah into cs.utah.edu, math.utah.edu, med.utah.edu, and law.utah.edu (which is further subdivided into www.law.utah.edu and ftp.law.utah.edu and a host of other specific Internet names for computers on the Law School network).
- **Communicating devices are peers.** Unlike other contemporary networks that divide computers into clients and servers (such as NetWare) or mainframes and terminals (such as SNA), TCP/IP treats every computer on the network as a peer—able to initiate or accept network connections independently of other computers (presuming, of course, that there is a network path between the two computers). Client and server software can be implemented on top of TCP/IP using sockets, but that is all irrelevant to the TCP and IP protocols. This means that TCP/IP is flexible and less likely to be vulnerable to failures of other computers that are not in the network path between the communicating computers.
- **TCP/IP is routable.** A routed network protocol makes it easy to pass data between two or more LANs or network links because routers simply retransmit the data in the payload portion of the network packet from one LAN onto another. Network protocols that can't be routed must rely on protocol gateways, which reinterpret the data on one network to allow it to conform to the addressing and data requirements of the other.
- **TCP/IP is independent of any particular transmitting medium.** TCP/IP will work over Ethernet, Token Ring, ARCnet, FDDI, USB, serial links, parallel port cables, short-wave radio (AX.25,) or any other mechanism that allows two or more computers to exchange signals. TCP/IP has even been defined to work using carrier pigeons as a packet delivery service!
- **TCP/IP is an open standard.** All of the documents describing the TCP/IP standard are available on the Internet for anyone to download and implement for free. There are no trade secrets or hidden implementation details limiting who may implement it.

- **TCP/IP is free.** TCP/IP was developed by universities with defense department funding, and anyone may implement it without paying royalties or licensing fees to any controlling body. Nobody "owns" TCP/IP. Or rather, everybody does.
- **TCP/IP is robust.** TCP/IP was designed when telecommunications lines between computers were not completely reliable, so the TCP/IP protocols will detect and correct transmission errors and gracefully recover from temporarily interrupted communications. TCP/IP will even route around damaged portions of the Internet.
- **TCP/IP is flexible.** TCP/IP is a protocol suite, with IP and a few other simple protocols at the bottom, and other protocols providing increasingly more sophisticated services layered on top. A simple network device, such as a router or print server, need only include those components required for it to do its job. Other, more complex devices, such as personal computers or domain name servers, implement a wider range of protocols to support their expanded functionality.
- **TCP/IP is pragmatic.** TCP/IP grew from a simple set of protocols. Additional protocols were added as the implementers found more uses for TCP/IP. This contrasts protocol suites designed ex-nihilo (such as the OSI stack), which, since nobody can think of everything, often leads to over-architected and brittle standards that don't quickly adapt to changing network requirements.
- **TCP is not perfect, however.** Two significant limitations are addressing and security. When it was first designed to link university and military computers, the implementers had no idea it would eventually grow to span the whole world. At the time, 32 bits of address space (allowing for approximately four billion computers) seemed plenty. Now, not only computers and routers, but also printers, terminal servers, scanners, cameras, fax machines, and even coffee pots connect to the Internet. Those 32 bits are being used up quickly, especially since address numbers are allocated in blocks and not all of the numbers in a block are actually used. Also (despite the military application of TCP/IP), the designers did not spend a great deal of effort securing TCP/IP against data snooping, connection hijacking, authentication attacks, or other network security threats. The era of electronic commerce lay too far in the future to worry about when they were designing a small communications system for a few elite researchers engaged in the open exchange of information.

So TCP/IP is cool, but how does it work? The next section will show you the nitty-gritty details of how your computer talks to those other computers on the Internet.

The Bit Bucket Brigade

Computer networks are complicated, and there is a lot you need to understand about TCP/IP in order to keep your network safe. Fortunately, you don't have to understand the whole structure of TCP/IP at once; you can start at the bottom of the stack (the TCP/IP suite is often called a protocol stack) where things are relatively simple, and work your way up. You can do this because TCP/IP is built in layers, each of which relies on the services provided by the layer below and provides more powerful services to the layer above. Figure 3.1 shows a graphical view of the layers in the TCP/IP protocol suite.

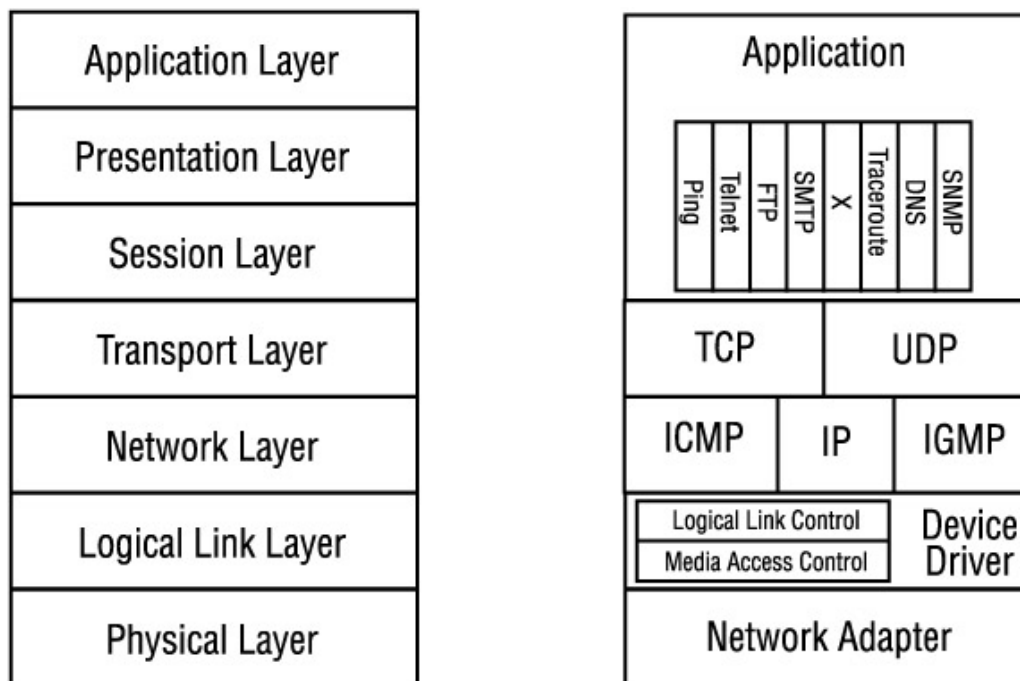


Figure 3.1: The TCP/IP protocol suite is composed of layers of services that roughly correspond to the layers of services defined in the OSI network model.

The International Standards Organization (ISO) has developed a useful model for comparing network protocols called OSI (Open Systems Interconnect). The OSI stack comprises seven layers, the first five of which describe the first five layers of the TCP/IP protocol suite. The bottom three layers of these first five describe how data transfers from one computer to another, and each is discussed in this section, starting at the bottom. The layers are traditionally numbered from bottom to top—therefore, the "Data Link" layer is "Layer 2."

Layer 1: Physical

Computer networking requires that each computer have a physical device (such as an Ethernet card or modem) to use to connect to the network. This device and the signaling characteristics of it, makes up the Physical Layer in the TCP/IP suite and the OSI stack. TCP/IP doesn't care what kind of device it is (TCP/IP is not dependent on any specific transmission medium, remember?), only that there is one and that data can be exchanged using it. TCP/IP relies on the operating system to configure and control the physical device.

Although TCP/IP doesn't care how the data physically gets from one place to another, you should. People trying to break into your network may chip away at any level of the network stack, including the Physical layer. You need to understand the security implications of each physical–network link choice in order to keep your network secure.

For convenience's sake, Physical–layer links can be divided into three categories based on connection behavior:

- **Dial-up** Temporary point-to-point connections over a shared infrastructure such as the telephone system
- **WAN and MAN (Wide Area Network and Metropolitan Area Network)** Constantly connected point-to-point connections
- **LAN (Local Area Network)** Two or more network devices communicating over a shared broadcast media

For each of the physical link options in each category we'll examine the security vulnerabilities and remedies for that option.

Dial-up

Dial-up connections are temporary; they are established when they are needed and reset at the end of the communications session. The biggest problem with dial-up communications (and digital leased lines as well) is that you cannot provide physical security at all points along the communications stream. The cables are run through the public infrastructure (under streets and over power lines) and other private establishments (the basement of your office complex, for example, where only janitors and telecom people dare to go).

Modem This communications medium uses regular, twisted-pair copper telephone lines for sending and receiving data and attaches to the phone lines just like a regular telephone. The modem modulates the outgoing serial digital signal into analog electrical signals in the same range as a telephone produces for human speech. It demodulates the incoming "tones" (actually just electrical signals corresponding to tones) back into serial digital bits for the computer to receive. Modem bit rates are typically low (up to 56Kbps).

- **Vulnerabilities** A physical tap on a phone line (either in the same building or at the phone company) can be fed into another pair of modems (one to receive each channel of the bi-directional communications), which can then demodulate the network traffic and feed it to an eavesdropping computer.
- **Remedies** Encrypt the data being sent over the modems.

ISDN This communications medium uses regular, twisted-pair copper telephone lines for sending and receiving data, but rather than converting to analog like a telephone, the data is sent digitally. Because ISDN does not connect to the phone wires like a regular telephone, the phone wires must be connected to a special, digital service. ISDN is provided in channels of 64Kbps, and the typical grade of services called Basic Rate is composed of 2 channels for an aggregate bit rate of 128Kbps. There is a lower speed ISDN channel bit rate for legacy circuits that operates at fast modem speed (56Kbps), and you can get up to 24 channels with Primary Rate, which operates at the same bit rate as a T1 circuit (1.5Mbps).

- **Vulnerabilities** As with a regular modem, a physical tap on a phone line (either in the same building or at the phone company) can be connected to a specially programmed ISDN modem, which can snoop on the network traffic and feed the intercepted communications to an eavesdropping computer.
- **Remedies** Encrypt the data being sent over ISDN.

WAN and MAN

WAN and MAN communications channels are typically links that are permanently maintained between locations, made either using the telephone infrastructure or wireless technologies such as radio, microwave, or lasers.

Dedicated Digital Leased Lines The most frequently used, permanent Internet connection for businesses today is a dedicated telephone line leased from the local phone company that is connected by a digital device called a CSU/DSU (Carrier Set Unit/Data Set Unit). These connections are like ISDN connections in that they are digital; however, they are not established and then shut down for each communications session as ISDN connections are, they are permanently connected. Also, the bit rate of a leased line ranges from modem speed (56 or 64Kbps

for a fractional T1) to many times faster than typical LANs (an OC12 allows 620Mbps). Leased lines may also be routed like a layer 3 network (as in the case of Frame Relay), but this routing is typically transparent to the customer (except in the case of X.25). See Figure 3.2 for a comparison of leased line data rates.

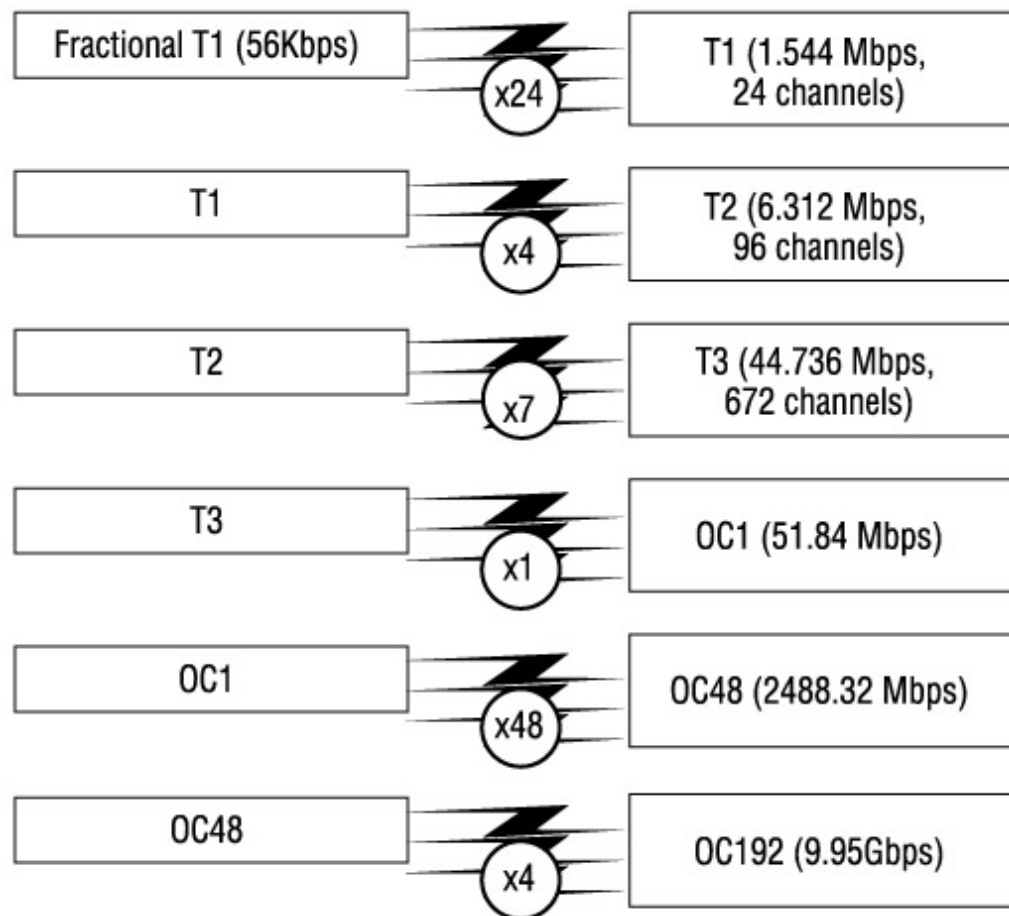


Figure 3.2: Leased line data rates range from 56Kbps all the way up to 2.5Gbps.

- **Vulnerabilities** As with a regular modem, a physical tap on a phone line (either in the same building or at the phone company) can be connected to a specially programmed DSU, which can snoop on the network traffic and feed it to an eavesdropping computer.
- **Remedies** Encrypt the data being sent over leased lines.

Radio, Microwave, and Laser Sometimes it is not feasible to run a physical cable between two locations. Islands, buildings separated by ravines, ships, and isolated communities, for example, need a way to exchange data without wires. NASA uses TCP/IP to communicate with some of its satellites, and for that application, copper cables are certainly not an option!

TCP/IP will operate just as effectively over a wireless medium as a wired one. The computer (or other network device) must, of course, have a transceiver for the medium—and there are transceivers for radio, microwave, and even laser communications. Most radio and microwave transmissions have stringent licensing requirements (there is only so much room in the RF spectrum, and government or military applications generally take priority), so there is a lot of paperwork as well as expensive equipment involved in setting up a radio or microwave link.

Warning The recent popularity of the 2Mbps 802.11, 11Mbps 802.11b, and 54Mbps 802.11a standards for wireless Ethernet means that radio will be deployed as the physical layer in and

between networks much more widely than it previously has been. The WEP (Wired Equivalent Privacy) encryption of the standard is weak and has been broken. If you install an 802.11 access point or bridge in your network you should treat it as an insecure medium and you should protect sensitive traffic flowing over it using other means.

- **Vulnerabilities** Broadcast media, such as radio and microwave, are even easier to eavesdrop on than cabled media. A single radio anywhere in the broadcast range of both the sender and the receiver of a radio link can eavesdrop on radio communications, while two receivers, each stationed behind and in the line-of-sight of the target transponders, can record the data being sent between them. Alternatively, two receivers directly between the transponders can eavesdrop on the communications, and since the power requirement is squared at twice that distance, the eavesdropping dishes can be much smaller. (Laser communications cannot be easily intercepted in this manner, but lasers are much more sensitive to environmental effects such as rain and snow.)
- **Remedies** Encrypt the data being sent over radio or microwave links. Consider using lasers for point-to-point communications in areas that are not adversely affected by weather and have adequate line-of-sight between communicating endpoints.

DSL This communications medium uses twisted-pair copper telephone lines for sending and receiving data, but they must be of sufficient quality and length to handle the greater voltages of the downstream DSL (Digital Subscriber Line) signal. Also, like ISDN, the data is sent digitally. Because DSL does not connect to the phone wires like a regular telephone, the phone wires must be connected to a special, digital service. DSL bit rates are much higher than regular modems (up to several Mbps depending on cable quality and filters).

- **Vulnerabilities** As with a regular modem, a physical tap on a phone line (either in the same building or at the phone company) can be connected to a specially programmed DSL modem, which can snoop on the network traffic and feed it to an eavesdropping computer.
- **Remedies** Encrypt the data being sent over DSL.

Cable Modems This communications medium uses the cable TV infrastructure for sending and receiving data. A portion of the cable broadband capacity is reserved for digital communications, and all of the customers in a neighborhood share that bandwidth like an Ethernet (the computer even connects to the cable modem using an Ethernet adapter). Cable modem bit rates are the highest of any low-cost Internet connection service (128Kbps upstream, up to 3Mbps downstream).

- **Vulnerabilities** As with Ethernet, any participant on the neighborhood network can sniff cable modem traffic. Cable modems are the least secure public transport for this reason.
- **Remedies** Encrypt the data being sent over cable modems.

LAN

While dial-up and WAN communications provide network links over large distances and generally connect just two computers together, LAN links are typically tied to a single physical location such as an office building and provide many computers with a shared communications medium.

Adequate site security can alleviate the problem of physical tapping of LAN communications, but when you develop the site security plan, keep LAN security requirements in mind.

Ethernet, Token Ring, FDDI, ARCnet, etc. Ethernet has become the glue that binds an organization together. Most organizations can still get some work done if the coffee pot breaks, the printer runs out of toner, or the Internet connection drops, but you can forget it if the network stops

working! Ethernet's speed, versatility, and ease of configuration have made it the LAN substrate of choice. From a hacker's point of view, however, all network types work similarly—cables are run to various locations, and computers are plugged into them. Any one computer on the LAN can transmit using electrical or optical signals to any other computer on the LAN. If a hacker can get control of one of the computers on the LAN, they can listen to all of the communicating computers.

- **Vulnerabilities** Any computer attached to a LAN segment can eavesdrop on all of the communication traversing it.
- **Remedies** Maintain strong physical security. If a portion of the LAN goes through a publicly accessible area (such as between buildings in a campus environment), consider using fiber optic cable for that section. Fiber optics are not easily tapped, and any break in the cable will terminate the link.

Serial Connections Sometimes you just need to link two devices, but you don't need a very fast connection—RS232 serial cables will do that just fine, and most computers come with serial ports built in. Serial cables make a good poor man's LAN, and serial cables have the same vulnerabilities that other LANs do.

- **Vulnerabilities** A serial cable can be spliced and the data sent over it fed to a third observing computer.
- **Remedies** Maintain strong physical security.

Layer 2: Data Link

At the very bottom of networking technology, signals are sent from one computer to another using an adapter (as the previous section shows, there are many kinds of signals and many kinds of adapters). But how does the computer talk to the device, and how are those signals organized into bits that the computer can make sense of? That's what the Data Link layer (Layer 2 in the OSI stack) is all about, and that's where the software meets the hardware.

Each networking adapter requires a piece of software, called a device driver, so that the operating system can control the hardware. The device driver must be tailored to the specific hardware device (such as an Ethernet card or FDDI adapter) that it drives. The operating system also requires a consistent way of simultaneously communicating with all of the network devices available to it. For this reason, the Data Link layer has been split (in the IEEE elaboration on the OSI network model) in to two sublayers:

- **The Media Access Control (MAC) Sublayer** Translates generic network requests (send and receive frames, device status, etc.) into device-specific terms.
- **The Logical Link Control (LLC) Sublayer** Provides the operating system link to the device driver.

Media Access Control

The MAC sublayer rests at the very bottom of the software stack, and does its work just before the hardware turns your data into electrical or optical signals to be sent out on the cable. This is the device driver, and it is responsible for controlling the hardware device, as follows:

- Reporting and setting the device status
- Packaging outgoing data received from the LLC sublayer in the format that the network adapter requires (in the case of Ethernet and PPP, a correctly constructed frame)
- Sending outgoing data at the appropriate time

- Receiving incoming data when it arrives
- Unpacking incoming data from the transmission format (i.e. the Ethernet or PPP frame), verifying the integrity of the data, and relaying the data up to the LLC sublayer

A network adapter actually receives all of the network frames transmitted over the link (if it is a shared media link, such as Ethernet) regardless of the intended destination because the network adapter has to read the recipient portion of the frame in order to determine if it is the intended recipient or not. The MAC sublayer discards all frames intended for some other recipient and only forwards data in frames intended for the MAC sublayer to the LLC sublayer above it.

The format of frames varies among link types, depending on the features supported by that networking technology. Ethernet, for example, has 48 bits of address space for identifying network devices, while ARCnet has only 8, and for PPP the addressing is irrelevant (the only device you can be talking to is the one at the other end of the line). Similarly, each supports a different data portion size, the ordering of status and control bytes differ, and some network types support features that others do not (such as compression, encryption, quality of service, authentication, and so on). Figure 3.3 compares Ethernet and PPP frames.

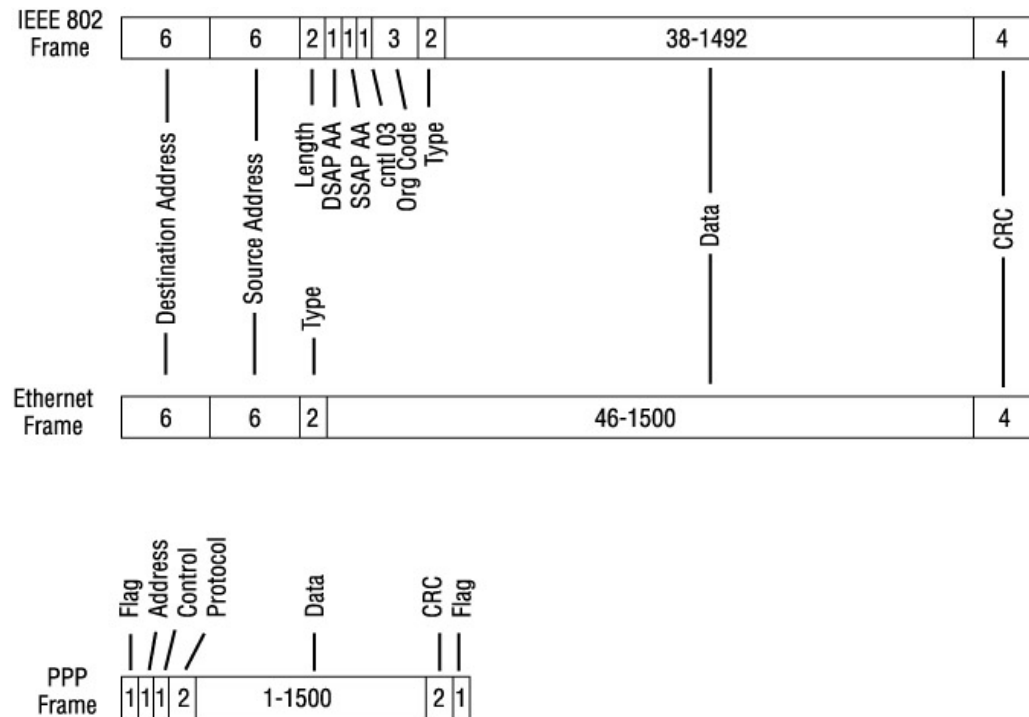


Figure 3.3: The structure of Ethernet and PPP frames are tailored to their uses (Ethernet for fast shared LANs, PPP for slow dial-up links).

Ethernet There are actually two frame types for Ethernet. The original Ethernet frame (defined in RFC 894) specified that the last two bytes indicate the type of the frame. The IEEE's reinterpretation of Ethernet (changed in order to fit it into their network taxonomy and defined in the IEEE 802.2 and 802.3 standards as well as in RFC 1042) uses the bytes at that offset as a length indicator. Fortunately, none of the RFC 894 types have the same two-byte value as valid IEEE 802 lengths, so network software can tell the two frame formats apart.

The fields the two frame types have in common are the six-byte address and data fields (giving 48 bits of hardware addressing) and the four bytes of cyclic redundancy check (CRC) at the end. For standard Ethernet frames (as opposed to IEEE 802.3 frames), a type of 0800 indicates that the data portion of the frame is an IP packet. 0806 is an ARP packet, and 8035 is a RARP request/reply packet. The IP packet can be from 46 to 1500 bytes in length, while the ARP and RARP packets are

28 bytes in length plus 18 bytes of padding, because the minimum data length for a standard Ethernet frame is 46 bytes.

For both kinds of Ethernet, those six-byte addresses identify the sender and the recipient in an Ethernet LAN. An Ethernet LAN is a network where the computers' communications are mediated only by hubs, switches, media converters, and bridges, not routers or firewalls. Ethernet cards are purchased with addresses pre-assigned to the cards (or to the device, for devices such as network printers that come with Ethernet built in). Because each hardware manufacturer is assigned a different range of Ethernet addresses to build into their devices, every Ethernet card or device should have a unique address. However many Ethernet adapters now allow their addresses to be over-ridden in software, so uniqueness is not guaranteed.

Warning Don't rely solely on unique Ethernet addresses to identify network frames from authorized computers. A network intruder could perform a denial-of-service attack on the authorized computer and bring up another compromised computer in its place on the network with the same Ethernet address configured in software.

Although the addresses in Ethernet frames are (or should be) globally unique, they can only be used to identify computers on the same Ethernet LAN. This is because the Ethernet frame contains no provisions for forwarding or routing between networks. Ethernet is a shared media network, in that every computer on it should be able to communicate directly with another device on the LAN without the Ethernet frame being reinterpreted and converted by an intervening router or firewall. While the frame may be selectively forwarded to other Ethernet segments and/or converted to new media by bridges and media converters, the actual contents of the frame must remain the same. Other LAN protocols, such as Token Ring, ARCnet, and FDDI have local addresses in their frames, not internetwork addresses that can be used to route data between LANs.

TCP/IP uses IP, ARP, and RARP to move data across the whole Internet, not just the local LAN. For now, you can just think of them as the data that has to be exchanged; from the Ethernet point of view, it doesn't matter what is contained in the data portion of the frame. Ethernet will convey other network protocols, such as IPX (used by NetWare,) EtherTalk (AppleTalk on Ethernet,) and NetBEUI (Microsoft's networking protocol) just as easily as it will convey TCP/IP.

Note We'll discuss IP, ARP, and RARP in more detail later on in this chapter.

For IEEE 802 frames, after the length field, there are three bytes containing 802.2 LLC information, and five bytes of SNAP information, the last two of which specify the type of data contained in the payload section. As with Standard Ethernet, a type value of 0800 specifies an IP datagram, 0806 specifies ARP, and 8035 specifies RARP. Because of the 8-byte LLC and SNAP overhead of IEEE 802 frames, the data portion of the frame may be from 38 to 1492 bytes in length, giving a maximum Ethernet packet a length of 1492 and ARP and RARP packets an absolute length of 28 bytes of data and 10 bytes of padding.

PPP The Point-to-Point Protocol was designed to support multiple network types over the same serial link, just as Ethernet supports multiple network types over the same LAN. It replaces an earlier protocol called SLIP (Serial Line Internet Protocol, which is still in wide use) that only supports IP over a serial link.

PPP frames have a five-byte header. The first three bytes are constant (7E FF 03 for the flag, address, and control bytes respectively), and the last two specify the protocol being transmitted in the data portion of that frame. The frame can hold up to 1500 bytes of data and is trailed by a

two-byte CRC and a one-byte flag (value 7E).

The three protocol types used by IP over PPP are the IP datagram with a protocol value of 0021, the link control data packet (C021), and network control data (8021).

Link Establishment Subversion There are several tricks that old-school telephone hackers (or phreakers, as they called themselves, they're a dying breed as the importance of telephones dwindles to nothing) can use to subvert or abuse dial-in networks. Call forwarding, for example, can re-route connection attempts to a hacker computer that can then eavesdrop on all your communications at the modem level rather than the IP level. If the computer making the connection supports several network protocols (newer versions of Windows can use IPX and NetBEUI as well as IP over PPP), the hacker can attempt to break in to the dial-in computer with one of these other protocols. Also, dial-up connections made over cellular phones (especially analog cellular phones; digital ones are a little more secure) can be eavesdropped on and even interrupted or captured by hackers with modified radios and cellular telephones.

Media Access Subversion Since it is up to the MAC to discard frames destined for other computers, hackers exploit this behavior by placing the device driver in *promiscuous mode* (replacing the device driver with one that supports promiscuous mode, if the installed device driver lacks it). Promiscuous mode simply relays all packets, regardless of their intended destination, to another program that monitors the data on your LAN.

Tip As a part of your greater security policy (beyond setting up and running a firewall), you'll want to monitor the computers on your network for changes to the device drivers, as well as to other important system files (such as the password lists and network service program files). You should also scan your network periodically for computers operating in promiscuous "sniffing" mode, using tools you can download from the Internet.

Logical Link Control

The LLC portion of the network stack is where the operating system sets up and controls the device driver as a general network device. If you have multiple Ethernet adapters, for example, you may have only one device driver but you will need several instances of it running, one for each Ethernet card. You may also have a Token Ring adapter and several serial port links in your computer, and the operating system will want to treat them all the same—as generic network devices that it can initialize, query the state of, deliver data to, and receive data from. Every operating system has a different specification for this layer, but all the device drivers for the operating system must meet the specification in order to operate as network devices. Windows has the Network Driver Interface Specification, (or NDIS) for example, and Unix has its character mode device specification (which, of course, varies among Unix implementations).

Layer 3: Network

TCP/IP doesn't specify how the Physical and Data Link layers work, it just expects them to provide enough functionality to link two or more computers together into a Local Area Network. That is, because TCP/IP is an Internetwork Protocol suite, it specifies how data can make its way from a computer on one LAN to another computer on a totally different LAN that could be as far away as half-way around the world or out of this world entirely.

TCP/IP does not replace other network technologies such as Ethernet or Token Ring. Instead, it incorporates their functionality and layers its own on top (this is in contrast to the supposed network

of the future—ATM—which attempts to provide one specification that works at all layers of the OSI stack). The next layer up in TCP/IP is the Internet Protocol, or IP.

Each layer in the stack exists to perform a specific function. The Data Link layer moves data across a LAN. The purpose of the Network Layer is to move data across as many LANs and network links as is necessary to get the data to its destination. IP performs this function well, and it performs only this function—other functions, such as ensuring that the data arrives in order and without duplication, or even that it arrives at all, are performed by other higher-level layers.

How Layering Works

Each layer in the OSI model (and in the TCP/IP suite) operates by using the data portion of the layer below it. IP over Ethernet, for example, places its data structures (called packets) in the data portion of the Ethernet frame. TCP (a layer above IP) places the communications streams it manages in the data portions of IP packets. Application level services such as FTP, which use TCP to establish and maintain the communications channels for exchanging files, write data to TCP sockets, which are placed in IP packets, which are placed in Ethernet (or PPP or Token Ring) frames to be sent out over the network link.

Another way of looking at the protocol stack is to start from the highest level and work your way down. As the data goes down through the TCP/IP networking layers, information specific to that layer is added to the data until it reaches the bottom, at which point it is sent out over the communications link (see Figure 3.4.). When it is received on the other side, the process is reversed, with each layer removing the data specific to that layer, until it is presented to the ultimate recipient of the data.

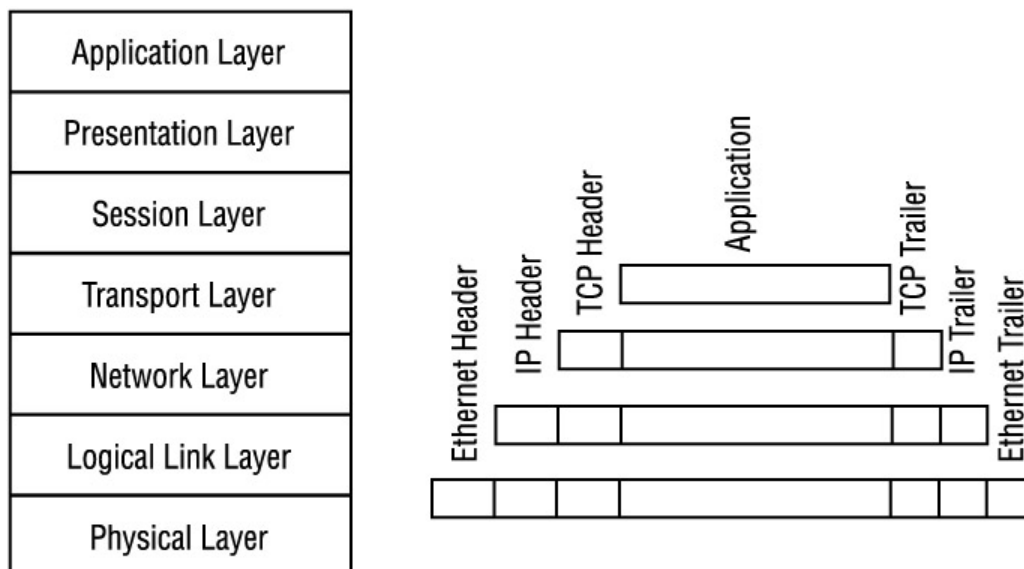


Figure 3.4: Each layer of the OSI stack adds layer-specific data to what it receives and passes the expanded information to the layer below it. When the layer receives information from the layer below it, the layer removes layer-specific data and passes the information on to the layer above it.

Frames and Packets The basic unit of Logical Link Layer data transmission is the frame. The Internet Protocol has a similar basic unit of data transmission—the packet. An IP Packet is quite similar in structure to an Ethernet frame, with source and destination addresses, packet description and option fields, checksums, and a data portion. Because of the way IP is layered on top of the Logical Link Layer, all of the packet structure is nested inside the data portion of the logical link frame (Ethernet, for example).

There are many different types of packets exchanged in a TCP/IP network, starting with the ARP and RARP packets (described in the "Machine vs. IP Addresses" section), and including IP packets (described in the next section of this chapter). For now, just understand that the Internet is based on packets nested in frames. The generic structure of an IP packet is illustrated in Figure 3.4.

The IP Header As shown in Figure 3.5, the IP header is typically 20 bytes long, but can be up to 60 bytes long if the packet includes IP options. The (non–optional) fields are as follows:

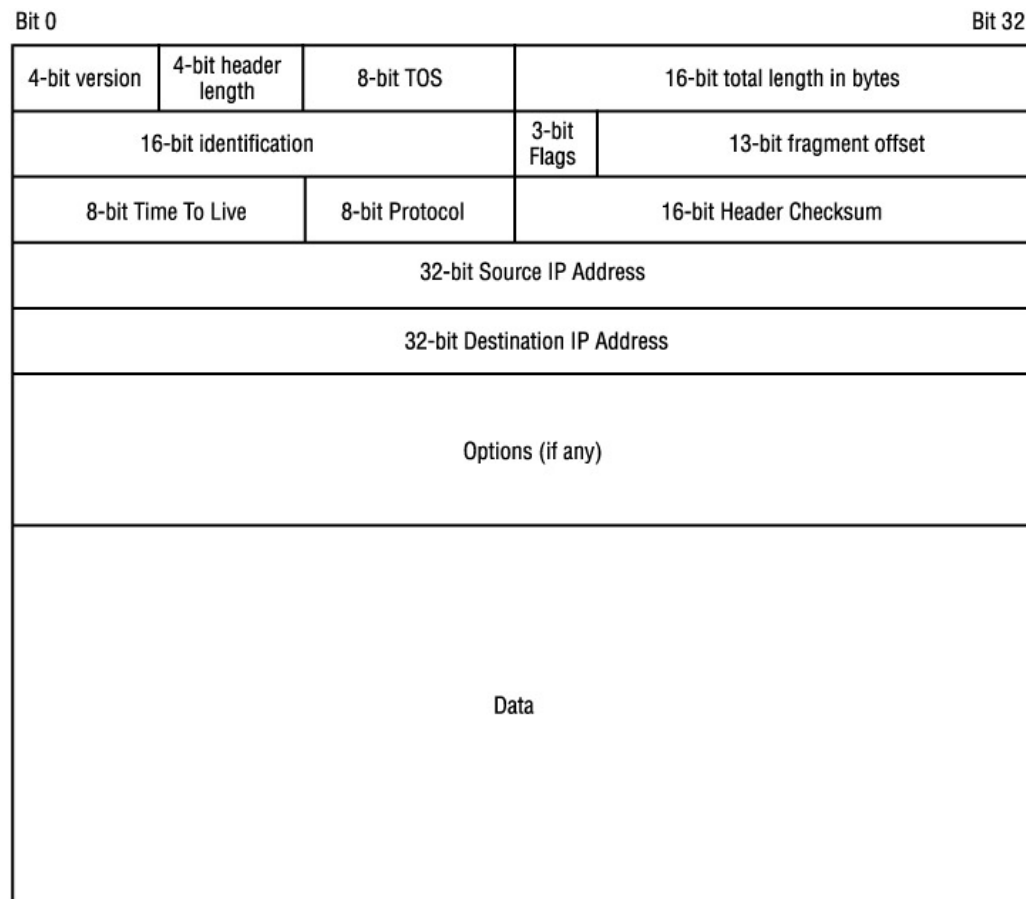


Figure 3.5: An IP packet has a header that includes the source and destination IP addresses, version, type, and service information, options, and a data section.

- **Version** These four bits identify which version of IP generated the packet. The current IP version is 4 for most users, although Tier–1 ISPs have deployed IP v6 on portions of their backbones.
- **Header length** This four–bit value is the number of 32–bit words (not bytes!) in the header, and by default its 20.
- **Precedence (TOS)** These eight bits are an early attempt at implementing quality of service for IP. They are composed of three bits for packet precedence (ignored by modern implementations of IP), four Type of Service bits, and a bit to be left at zero. Only one of the four Type of Service bits can be turned on. The four bits are: minimize delay, maximize throughput, maximize reliability, and minimize cost. All zeros mean the network should use normal priority in processing the packet. RFCs 1340 and 1349 specify TOS use. Most implementations of IP don't allow the application to set the TOS value for the communicated data, which limits the usefulness of this field.
- **Datagram Length** This is the total length of the IP datagram in bytes. Since this is a 16–bit field, the maximum IP packet size is 165535 bytes in length, even if the Data Link layer frame could accommodate a larger packet.

- **Identification** To guarantee that packets are unique and to assist in the reassembly of data streams that have been broken down into packets, the sending computer numbers the packets, incrementing the value when each packet is sent. This value doesn't have to start at zero, and isn't necessarily incremented by one on all implementations of IP.
- **Flags** Flags define whether or not this packet is a fragment of a larger packet.
- **Fragment Offset** This defines where in the chain of fragments this fragment belongs when reassembling a fragmented packet.
- **Time To Live** This declares the number of routers through which the packet may pass before it is dropped. This prevents packets from getting caught in routing loops. This field is typically set to 32 or 64. When a packet is dropped due to TTL expiring, an ICMP message is sent to the sender.
- **Protocol** This field shows which IP protocol (TCP, UDP, ICMP, etc.) generated the packet and should be the recipient of it at the destination.
- **Header Checksum** All of the 16-bit values in the packet header are summed together using one value's complement addition and placed in this field.
- **Source** This is the computer or device sending the IP packet.
- **Destination** This is the intended ultimate recipient of the IP packet.

Network Byte Order (Big-Endian)

Internet Protocol packets are sent in Network Byte Order, where the most significant bit of a word is sent first. For example, the 32-bit word 0xFFE0A102 would have the byte 0xFF sent first, and the byte 0x02 sent last. For this reason, Network byte order is also called *big-endian*. The other way of doing things is called *little-endian*, where the least significant bits are sent first.

Network Byte Order can be a little confusing to think about. If you number the bits in a word being sent, starting with zero, the most significant bit is the 0th bit and the least significant bit is the 31st bit. This is the opposite of how computer professionals are used to numbering bits.

Of course, regular people (and computer professionals, too) like to count forwards rather than backwards when sending and receiving things, and there are other good reasons to send data in big-endian order. Plus, the Internet requires a standard byte order. Therefore, Most Significant Bit first, starting the count at zero, is the way to go.

Networks of Networks

The Internet is more than just a really big LAN—it is a network of networks, in much the same way a LAN is a network of individual computers. And just as every computer on an Ethernet has an Ethernet address, every network in the Internet has a network address. In addition, every computer on the Internet also has a station address identifying it on that network. Together, the network address and the station address form the IP address, uniquely identifying that computer on the Internet.

Networks vary in size, from small home or office networks to multinational corporations and large ISPs that may have thousands of computers connected to their network. There are many more smaller networks than there are large ones. The designers of the Internet Protocol took advantage of this fact by making the sizes of the network and station addresses variable, with the network portion getting larger as the station portion gets smaller. That way, the address size as a whole can remain constant. With the eternal optimism of computer programmers everywhere, they decided that 32 bits would be plenty—after all, what was the likelihood that anyone would want to connect

more than four billion devices to this little academic network they'd created?

Class and Classless Addressing

It used to be that you could tell the size of a network by the first byte of the network address. If the first bit was zero (i.e., if the byte was from 0 to 127), the address was a class A network. If the first bit was one and the second was zero (if the byte was from 128 to 191), it was a class B network. If the first two bits were one and the third was zero (if it was from 192 to 223), it was a class C network. So, a simple look at the IP address told you how big your LAN was and whether or not another computer was on the same network. (It was on the same network if it had identical network address bits as specified by the class, and a different set of station bits in the IP address.)

But things have changed. With the advent of classless addressing, you can no longer expect your IP address to reflect the size of the network the computer is on. Instead, you have to examine the subnet mask, which explicitly defines the size of the network address portion and the station address portion of the IP address. Classless addressing started as an easy way for organizations with large networks to efficiently break up their large IP ranges into subnets (that's why it's called a subnet mask), and as the IP address range has filled up, classless addressing has proven itself a useful tool for scavenging smaller class C network numbers out of the large class B and class A spaces.

If you were paying attention, you'd have noticed that classes A, B, and C did not fill up all 32 bits of IP address space. There were two additional classes specified, as follows:

- Class D (with the first three bits set to one and the fourth set to zero), which was reserved for IP multicast (sending packets to multiple networks)
- Class E (with the first four bits set to one and the fifth set to zero), which was reserved for future use

Until address space started to run out, an organization could request a range of addresses in one of three classes that conveniently mapped to byte boundaries, as follows:

- **Class A** There were only 125 (0, 10, and 127 reserved) of these, because only seven bits were used for the network address (the first bit was zero) and several network addresses are reserved, but a class A network could have more than sixteen million stations.
- **Class B** There were about 16,000 of these, each of which could have more than 65,000 stations, because the network portion of the address was 14 bits and station portions of the address were 16 bits in size.
- **Class C** Almost the opposite of a class A network, there were about two million possible class C networks, each of which could have up to 254 station addresses.

The number of networks and stations are always a little less than the number of bits allocated to the network size because several network and station addresses are reserved for special functions. The lowest and highest address (0 and 255 for class C, 0 and 65536 for class B, etc.) are used for broadcasting to all of the computers in the local network, for example.

So, when an organization wanted to connect to the Internet they got a range of IP addresses sufficient for the number of computers they wanted to hook up. Internally, the computers all shared the same network number, but each had different station addresses.

Machine vs. IP Addresses

If Ethernet addresses are globally unique, why not use them as the Internet address? Ethernet addresses could not be used as a global addressing scheme because a) not every computer has an Ethernet adapter, and b) the sending computer needs to know not only the recipient's address, but also how to send the data to that address (there's no telling where in the world any particular Ethernet adapter might end up).

For this reason, IP uses its own addressing scheme in addition to whatever addressing scheme is used by the network adapter in the machine. IP addresses are four bytes long, and (unlike Ethernet addresses) they are hierarchical in nature. While Ethernet cards from different vendors (and therefore with completely different Ethernet addresses) can be mixed with impunity in a LAN, all of the TCP/IP-capable computers that communicate directly with each other (i.e., that are in the same subdomain) have anywhere from 3 to 30 bits of the address in common, depending on the size of the subdomain the computers participate in.

The Ethernet, Token Ring, ARCnet, FDDI, or AX.25 address (depending on the Data Link layer media connecting the computer to the Internet) is still used to get the IP data across the LAN. To avoid confusion, the Data Link layer address is called the Machine Address, while internetwork addressing is done using an IP address, which is more than just a number.

Network Addresses, Subnet Masks, and Station Addresses As described in the previous section, every IP address has two parts: those bits it has in common with the other computers in its subdomain (called the network address) and those bits that are unique for computers in the subdomain (the station address). The bits for the network address always come before the bits for the station address. A third IP configuration value, the subnet mask, tells the computer which part of the IP address is the network address and which part is the station address, as shown in Figure 3.6.

Decimal	Binary			
IP Address	MSB			LSB
128.110.121.213	10000000	01101110	01111001	11010101
Subnet Mask				
255.255.255.192	11111111	11111111	11111111	11000000
Network Address				
128.110.121.192	10000000	01101110	01111001	11000000
Station Address				
21				010101

Figure 3.6: The subnet mask identifies which part of the IP address is the network address and which part is the station address.

All of the computers that have the same network address are expected to be able to directly communicate with each other using the Data Link layer. All of the computers on one Ethernet LAN could be put in one subnet, for example, and all of the computers in a Token Ring LAN could be put in another. Mixed Token Ring and Ethernet computers can't all be put in the same IP subdomain (i.e., they could not be given the same network address), because an Ethernet-enabled computer could not send an Ethernet frame directly to a Token Ring-enabled computer.

Every subnet that will be connected to a larger network, such as a campus network or the Internet, must have at least one machine that has multiple network adapters so that it can move data between the LAN and the exterior network. This machine is called the gateway in Internet terminology (see Figure 3.7), and can be a special purpose adapter (such as a fast hardware router) or a general purpose computer that routes network data as well as performing other functions (such as firewalling and hosting FTP and web services). This machine will have two (or more) IP addresses, one for the network adapter residing on the local network and one (or more) for the network adapter(s) residing on the external one(s).

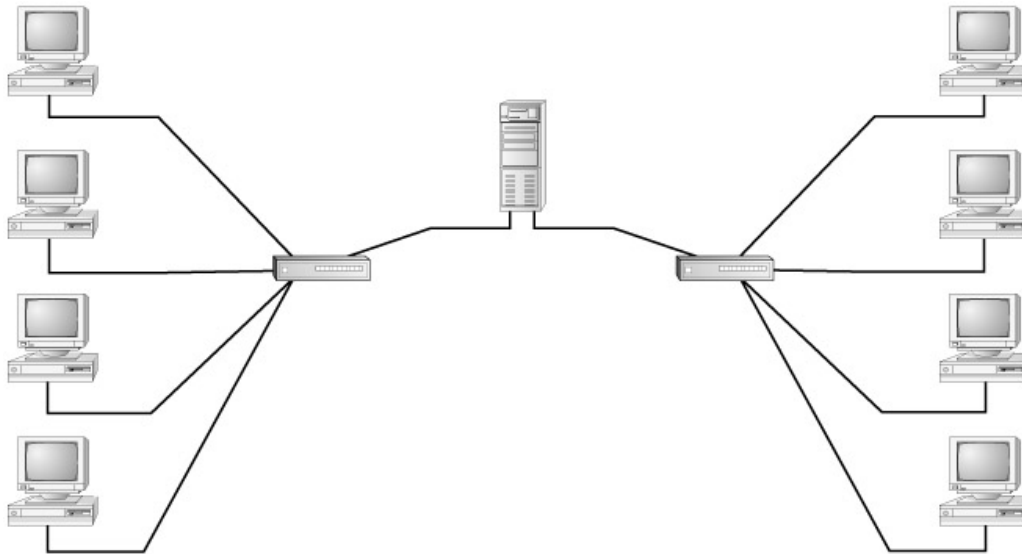


Figure 3.7: A gateway machine has two network adapters and allows network traffic to move between LANs.

A multi-homed host may reside on networks of different sizes. Therefore, each network adapter (Ethernet, CSU/DSU, modem, etc.) has a different subnet mask associated with its IP address, identifying (for that IP address only) which part of it is the network address and which part is the station address for that network adapter.

Communicating with a Local Host Sending IP data is a little more complicated than sending Ethernet data, partially because IP uses Data Link protocols such as Ethernet to do part of its job. With Ethernet, a computer that has data to send just broadcasts it. Every computer on the LAN then receives it, and (unless a hacker has put an adapter in promiscuous mode) only the recipient keeps it. IP, on the other hand, needs first to determine whether or not the destination computer is on the local network, and if so, what its corresponding machine address is before it can send the packet embedded in a Data Link frame.

IP operates simply to determine if the destination computer can be communicated with directly. For each of the network devices attached to the computer, the computer compares the network portion of the adapter IP address (as identified by the subnet mask for that adapter) with the same bits (but only those bits, as identified by that same subnet mask) of the destination IP address. If the bits match, the destination computer resides on the subnet that adapter is connected to, and the computer can send the data directly to that computer using that adapter.

But the Data Link layer, which IP uses to actually send the data, doesn't know anything about IP addresses, so IP needs to get the Machine Address of the destination computer on that LAN before it can put the IP data into a Data Link layer frame and send it out. The Internet Protocol uses another protocol, called the Address Resolution Protocol (ARP), which every computer on an IP subnet hears and can respond to, to translate IP addresses into Machine Addresses.

ARP works by filling a Data Link layer frame with a special packet that every computer on the LAN will receive. As you saw in the Data Link layer section, the frame type for Ethernet for an ARP packet is 0806. Figure 3.8 shows the structure of an ARP packet (which is contained in the data portion of the Data Link layer frame).

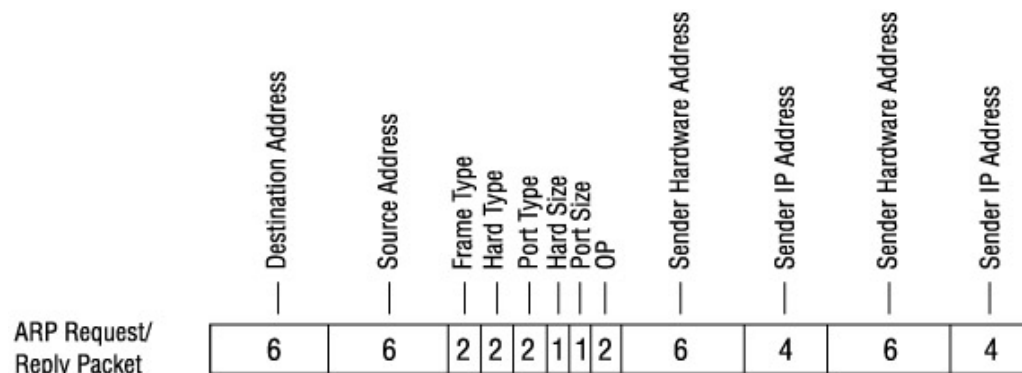


Figure 3.8: An ARP/RARP packet is a broadcast request for data.

The fields for an ARP/RARP packet are defined as follows (this example is based on Ethernet; other Data Link layer packets may have different sized values):

Field	Definition
Hardware Type (two bytes)	The type of hardware address (1 for Ethernet)
Protocol Type (two bytes)	The protocol type (0x0800 for Internet Protocol)
Hardware Size (one byte)	The size of the hardware addresses in bytes
Protocol Size (one byte)	The size of the protocol addresses in bytes
OP (two bytes)	The type of operation: either ARP Request, ARP Reply, RARP Request, or RARP Reply
Sender Hardware Address (six bytes for Ethernet)	The Ethernet address of the ARP packet sender
Sender Protocol Address (four bytes for IP)	The IP address of the ARP packet sender
Target Hardware Address (six bytes for Ethernet)	The Ethernet address of the ARP packet sender
Target Protocol Address (four bytes for IP)	The IP address of the ARP packet sender

To get the hardware address of a computer in a local subnet, a computer sends out (in a broadcast Data Link layer frame) an ARP packet with an OP value of 1 and everything but the Target Hardware Address filled in (because only the sender needs to know that). Since the frame is broadcast to all stations on the LAN, all of the computers will receive it, but only the one with the matching Target Protocol Address will respond by changing the OP value of the packet to 2, filling in the Target Hardware Address, and broadcasting the packet back.

The computer does not perform an ARP for every IP packet it sends out. The computer maintains a table of IP addresses and corresponding hardware addresses, and only when the computer does not have an entry for an IP address in its subnet does it send out an ARP packet and wait for a reply. The rest of the time it just uses the value it has in its ARP table.

When the computer has the hardware address corresponding to an IP address for a packet, the computer sends the IP packet to the destination in a Data Link layer frame addressed to just that

computer. That computer receives the frame, notes that the destination IP address is its own, and processes the packet according to the protocol information in the header.

Note Why does IP bother with ARP and machine addressing when Ethernet is a broadcast medium anyway? After all, every Ethernet adapter in the LAN can listen to all the packets, not just ones with its own Machine Address as the intended recipient. Why not just send all of the IP packets as broadcast frames, and let the IP protocol on each computer sort them out? The reason is two-fold: Ethernet hardware can be much more efficient than the IP protocols at determining which frames should be ignored and which ones should be processed, and Ethernet networks are often partitioned into many small collision domains by bridges and switches in order to decrease contention and increase available bandwidth. Broadcasting defeats this optimization by spilling traffic over into collision domains that contain neither the sender nor the recipient.

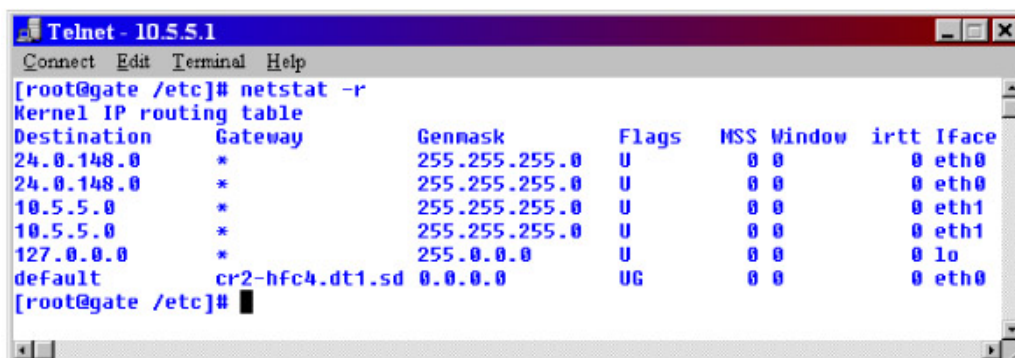
Communicating with a Distant Host But what if the destination IP address does not lie on any LAN the computer is connected to? The computer could send out an ARP message, but nobody would reply.

If none of the network addresses of any of the computer's adapters exactly match the destination IP address (as defined by the subnet mask for that adapter), then the computer needs to know where to send the packet so that it will eventually get to its destination. Since it can't send the packet directly to the destination, it needs to know where else it can send the packet that will eventually get it there.

For most networks, that intermediate location is the gateway host or router. As described in the section above on network addresses, subnet masks, and station addresses, the gateway is a machine that has adapters residing on at least two networks. Most LANs connected to the Internet will have one gateway, but some may have more than one. Just one of them should be configured to be the default gateway, where computers in the LAN send an IP packet when they don't otherwise know what to do with it.

All your standard computer needs then is a default gateway entry to send non-local packets to. The gateway can then figure out where to send the packet from there. This means, of course, that the gateway requires a bit more configuration than computers with just one network adapter. The configuration information goes in a routing table that tells the router what to do with those packets it receives that have an ultimate destination of somewhere else.

A routing table is simply a list of station and network addresses and a corresponding gateway IP address further along to which it should send packets that match that station or network address. Figure 3.9 shows a sample routing table.



The screenshot shows a Telnet window titled 'Telnet - 10.5.5.1' with a menu bar (Connect, Edit, Terminal, Help). The command '[root@gate /etc]# netstat -r' has been executed, displaying the 'Kernel IP routing table'. The table lists destinations, gateways, netmasks, flags, MSS, window, irtt, and interfaces.

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
24.0.148.0	*	255.255.255.0	U	0 0		0	eth0
24.0.148.0	*	255.255.255.0	U	0 0		0	eth0
10.5.5.0	*	255.255.255.0	U	0 0		0	eth1
10.5.5.0	*	255.255.255.0	U	0 0		0	eth1
127.0.0.0	*	255.0.0.0	U	0 0		0	lo
default	cr2-hfc4.dt1.sd	0.0.0.0	UG	0 0		0	eth0

Figure 3.9: The routing table instructs the gateway how to forward IP packets.

The simplest gateway configuration requires just three entries—one for itself (127.0.0.1 is IP shorthand for the computer itself), one for IP addresses in the interior LAN, and one for all other IP addresses (its own default gateway when it doesn't know what to do with an IP address).

Things get a little more complicated when you have a more complex network structure behind your gateway, but the principle remains the same—the routing table explicitly lists all of the networks behind the gateway and where the next closest (internal) gateway is for those internal networks, and the routing table lists a default gateway entry to send packets to that the gateway otherwise doesn't know what to do with. See Figure 3.10 for a more complex routing example.

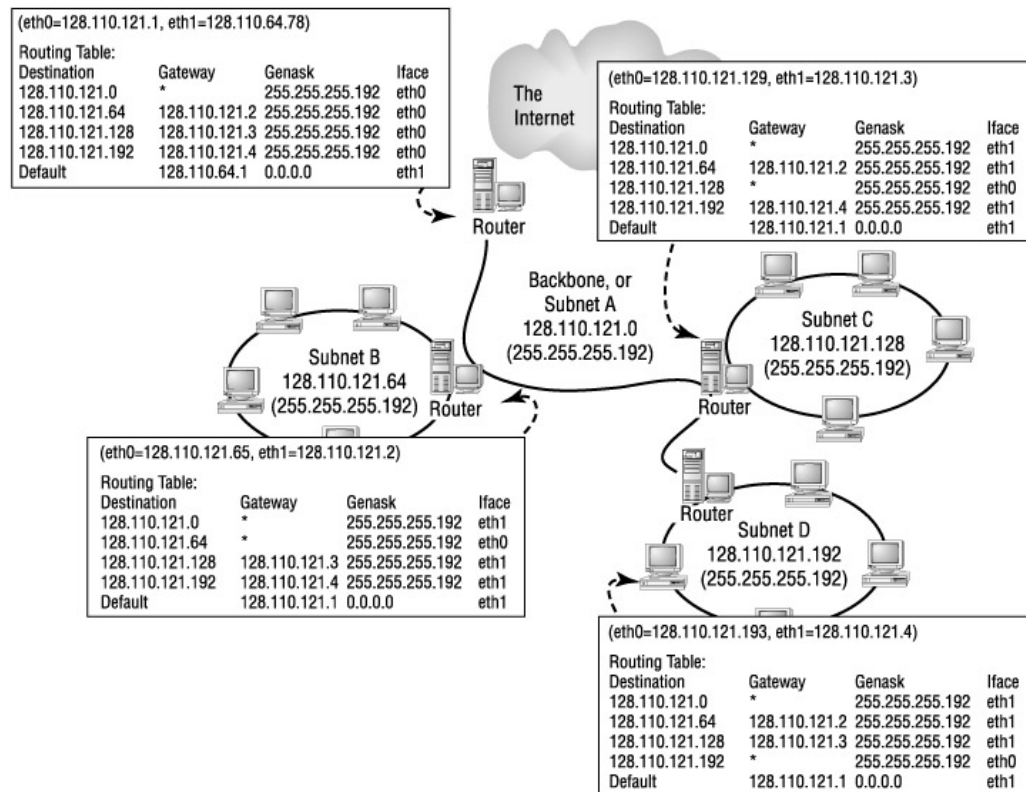


Figure 3.10: A gateway should know how to explicitly route to any internal network and also have a default external gateway to send packets it doesn't know what to do with otherwise.

A gateway or router processes a packet as follows (see Figure 3.10 for a graphical view):

1. Accept Data Link layer frame.
2. Verify CRC strip frame header and trailer; discard if invalid.
3. Verify IP header checksum; discard if invalid.
4. Check IP parameters; discard if invalid or does not match policy.
5. Decrease the Time to Live field in the packet. If the field now reads 0, discard the packet and send an ICMP message back to the sender of the packet.
6. Check the destination IP address against the router's own IP addresses; if there is a match, process the IP packet locally.
7. Find the network or host entry in the routing table that matches the destination IP address of the packet. If there is no match, use the default entry in the routing table.
8. Create a new Data Link layer frame for the packet with the hardware address of the next closest gateway to the destination (or the hardware address of the destination itself if it is on the same LAN).
9. Send the Data Link layer frame.

The nice thing about IP routing is that no single computer needs to know how to get the IP packet all the way to the destination—all each computer needs to know is how to get the packet to the next computer on the way. So long as there are no problems along the way, the packet will eventually get to its destination.

ICMP Messages

But what if there is a problem? Let's say, for example, that the default network link for your connection to the Internet has gone down, and (while the network technicians frantically try to get it up) an alternate host has been provided to route data to and from the Internet.

The gateway that would otherwise forward your IP packet to its destination needs a way of telling your computer about the network problem. ICMP packets are the Internet Protocol's way of sending network status information back to the sender. With ICMP, network devices can identify routing problems, congestion problems, quality of service problems, and more. Table 3.1 shows the types of ICMP messages your computer can send or receive.

Table 3.1: ICMP Message with general message type, specific message code, and whether the message is a query or a response to a network event

Type	Code	Description	Q/E
0	0	Echo reply	Q
3		Destination Unreachable:	
	0	Network Unreachable	E
	1	Host Unreachable	E
	2	Protocol Unreachable	E
	3	Port Unreachable	E
	4	Too Big/Can't Defrag	E
	5	Source Route Failed	E
	6	Destination Network Unknown	E
	7	Destination Host Unknown	E
	8	Source Host Isolated (obsolete)	E
	9	Destination Network Prohibited	E
	10	Destination Host Prohibited	E
	11	Network Unreachable for TOS	E
	12	Host Unreachable for TOS	E
	13	Prohibited by Filtering	E
	14	Host Precedence Violation	E
	15	Precedence Cutoff in Effect	E
4	0	Source Quench	E
5		Redirect	
	0	Redirect for Network	E
	1	Redirect for Host	E
	2	Redirect for TOS and Network	E
	3	Redirect for TOS and Host	E
8	0	Echo Request	Q
9	0	Router Advertisement	Q

10	0	Router Solicitation	Q
11		Time Exceeded	
	0	TTL Exceeds 0 During Transit	E
	1	TTL Exceeds 0 During Reassembly	E
12		Parameter Problem	
	0	IP Header Bad	E
	1	Required Option Missing	E
13	0	Timestamp Request	Q
14	0	Timestamp Reply	Q
15	0	Information Request (obsolete)	Q
16	0	Information Request Reply (obsolete)	Q
17	0	Address Mask Request	Q
18	0	Address Mask Reply	Q

IP identifies by the Protocol field what kind of packet is being transmitted (see Figure 3.5). ICMP is one of those kinds of packets. The structure of an ICMP packet is shown in Figure 3.11.

Destination Unreachable

Type 3 (8)	Code (8)	Checksum (16)
Unused (16)		Next Hop MTU (16)
Internet Header + 8 bytes of foiled datagram		

Echo Request or Reply

Type 8/0 (8)	Code (8)	Checksum (16)
Identifier (16)		Sequence # (16)
Data		

Time Exceeded

Type 11 (8)	Code (8)	Checksum (16)
Unused (16)		
Internet Header + 8 bytes of foiled datagram		

Address Mask

17/18 (8)	Code (8)	Checksum (16)
Identifier (16)		Sequence # (16)
Address Mask		

Source Quench

Type 4 (8)	Code (8)	Checksum (16)
Unused (16)		
Internet Header + 8 bytes of foiled datagram		

Timestamp Request/Reply

13/14 (8)	Code (8)	Checksum (16)
Identifier (16)		Sequence # (16)
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

Redirect

Type 5 (8)	Code (8)	Checksum (16)
Address of Router to be used (16)		
Internet Header + 8 bytes of foiled datagram		

Destination Unreachable

Type 12 (8)	Code (8)	Checksum (16)
Pointer (16)	Unused (16)	
Internet Header + 8 bytes of foiled datagram		

Figure 3.11: CMP messages share a common header format with Type, Code, and Checksum fields.

So what happens when your computer tries to send a packet through the obstructed gateway? The steps that the gateway and your computer go through are as follows:

1. Your computer sends a packet with the frame address of the gateway machine, but with the IP address of an exterior computer.
2. The gateway machine reads the packet from the frame and decodes it.
3. The gateway machine discards the packet (because the packet cannot be forwarded) and creates an ICMP message.

4. If the gateway has not been configured to redirect traffic to an alternate gateway, it will send a host-unreachable or network-unreachable ICMP packet back to the sending computer.
5. If the gateway has been configured to redirect traffic, it will send an ICMP redirect packet to the sending computer.
6. The sending computer receives the ICMP packet and decodes it.
7. If the packet is a host- or network-unreachable packet, it generates an error message for the program that attempted to send the data.
8. If the packet is an ICMP redirect packet, it resends the data using a different gateway.

This sort of network reporting and redirection can happen at any point in the string of packet "hops" from the sending computer to the receiving computer. The operations performed are the same.

ICMP Vulnerabilities

ICMP packets do not include any authentication method for the recipient of the message. A clever hacker can forge ICMP packets and cause havoc in an unprepared network. The two greatest threats from malicious ICMP packets are denial-of-service attacks and impersonation, or man-in-the-middle, attacks.

A forged destination-unreachable packet can isolate a computer from necessary services. Echo Request has been used by hackers to crash computers that have a naive implementation of the ICMP protocols.

Once a computer that performs an important service (such as a DNS server, file server, or web server) is out of the way, an ICMP redirect packet can point unwitting victims to the hacker's computer, where the hacker can accept authentication information (usernames, passwords, etc.).

You can guard against these threats by configuring your firewall to discard ICMP packets that change network behavior (such as ICMP Redirect) and that have come from untrusted sources (such as anywhere outside your own network.)

Routers and Route Discovery

Routers (sometimes called gateways) link IP subnets together. It is up to the router to send IP packets toward their destinations when the end computer resides outside of the subnet. How does the router know where to send the packet?

For small networks with just one or two routers the answer is easy—the network administrator enters routes by hand into the router. This is called static routing because the route information does not change automatically according to network conditions—it is up to the administrator to change the router's settings when a new subnet is added or a network link goes down.

The router must have IP forwarding rules that account for any possible IP address because there's no telling which computer, locally or on the Internet, a client computer may want to talk to. On a simple network the rules are correspondingly simple. On a one-router network, for example, the rule may be as follows: If the destination address is on the local subnet, then send the packet directly to its destination, otherwise send it to an upstream router.

Dynamic Routing In larger networks, the static routing tables get to be a real hassle to maintain. In a university with every college maintaining its own LAN, or a corporation with each department

having a different subnet, the configuration of the network is always changing and the router tables must be updated to reflect those changes. In addition, the routers must have a consistent view of the network.

The IP routing protocols reduce the work required to keep those routing tables updated and consistent. The protocols enable the routers to automatically update each other of network changes such as new routes between subnets or network links that are down and must be routed around. The administrator makes a change in just one router's table, and the change will propagate automatically to all of the other routers in the network.

Interior Gateways vs. Exterior Gateways Before you take a closer look at the routing protocols, you need to understand that there are two categories of routers—interior routers (or gateways) that reside entirely within your organization, and exterior gateways that convey IP traffic between your network and the Internet (or any other organization's network). The difference between these two types of gateways is primarily a matter of trust; you want routing changes within your network to move from one router to another with minimum hassle, while you may not want routing information from outside your network to be implemented on your border routers unless it comes from an authorized source and until you've had a chance to review it first.

RIP v1 and v2, IGRP The earliest routing protocol in widespread use is called (imaginatively enough) Routing Internet Protocol, or RIP. Routers using version 1 of the protocol simply broadcast their entire routing table every 30 seconds on each of the subnets to which that router has network adapters connected. When a router receives a RIP packet, it compares the contents with its own table and updates its table with new information from the packet (modified, of course, to include the extra hop that it takes to get to the router that sent the RIP packet). It removes routes that haven't been updated for a while, so bad routes eventually disappear from the table.

RIP is a distance–vector protocol because it keeps track of the distance (number of network hops) that it takes to get to a destination listed in the routing table. When multiple routes to the same destination show up in the table, the route with the least number of hops is used to send the packet on its way, regardless of the speed of the network links used in the route. RIP packets are broadcast using UDP.

RIP v1 had a significant security problem—there was no authentication included in transmitted RIP packets, and received RIP packets weren't checked to make sure they came from authorized routers. Anybody could forge a RIP packet and reconfigure your router for you. RIP also has the problem that in complex networks with large routing tables and lots of routers, the RIP broadcasts can soak up a lot of network bandwidth. Also, it can take a long time for network changes to propagate through all of the routers using RIP in a complex network.

RIP v2 fixes the authentication problem by including passwords in the RIP packets. The router discards any RIP packets that don't have a correct password. The bandwidth and convergence delay problems of RIP remain, however.

For their high–performance routing hardware, Cisco developed a new protocol called IGRP, which was endorsed by the IETF. IGRP maintains more link information, such as bit rate and cost, allowing the routers to make better decisions about possible routes. IGRP has a 90–second update interval, however, and it can take even longer than with RIP for IGRP routers to converge on a consistent set of routing tables.

OSPF In 1988 the Internet Engineering Task Force (IETF) came out with a protocol designed to replace RIP for routing within a single organization's network (i.e., an IGP). That protocol was called

Open Shortest Path First, or OSPF. It is a more complicated protocol than RIP, but has reduced network bandwidth use and speedier convergence. This makes RIP a much better protocol for routing in medium-to-large networks.

OSPF is a link-state protocol, unlike RIP (which is a distance-vector protocol). OSPF records in its routing table the state of the link between it and any other routers it can directly connect to, along with those IP domains that its network adapters reside on. In addition to status, it records link characteristics such as bandwidth, cost, and type-of-service restrictions. OSPF uses IP multicast as well as broadcast for transmitting routing table updates. It also performs load balancing, which sends successive IP packets along different routes to the same destination to reduce congestion on the network.

Route updates in an OSPF routed network propagate much more quickly than they do in a RIP routed network. After the initial update, only the difference between one update and the next is transmitted. This is unlike RIP, which transmits the entire table each update. Also, OSPF imposes a hierarchy on the routers in the network, with Designated Routers for each subnet, a Backbone Router for the network backbone, and Autonomous System Boundary Routers for links between separately administered systems. The hierarchy makes administration a little more difficult, but the centralized collection and redistribution of route data allows the routers to converge on a consistent view of the network in a matter of seconds rather than minutes or hours for a large network.

OSPF, like RIP v2, authenticates the routing packets exchanged between the routers, making it much more difficult for network intruders to subvert them. In addition, the OSPF packets contain sequence numbers, making forgery difficult even if the authentication mechanism is compromised.

Note You should block all RIP, IGRP, and OSPF traffic at the firewall. Because your firewall will be the only connection to the Internet (otherwise, why have a firewall?), you have no need to maintain route information from the Internet on your internal routers. The only Internet configuration your routers need is that IP packets with destinations exterior to your entire protected network should go to the firewall.

BGP and IDRP Unless you run the network for an Internet service provider or Network Access Point, you don't need to worry about Exterior (or Border) Gateway Protocols. We'll touch on them here, however, for the sake of completeness.

The original Border Gateway Protocol was designed for the routers at the edges of private networks. These routers must maintain huge routing tables because the Internet is a large place, and there's no telling where on the Internet an IP subnet is going to be until the router containing the subnet announces the fact and the route is propagated across the public network.

BGP is a distance-vector protocol because it maintains hops like RIP. However unlike RIP, BGP uses TCP rather than UDP to exchange routing information.

With the expansive growth of the Internet, BGP began to show its age and its naive security assumptions. The IETF developed an Inter Domain Routing Protocol (IDRP) to replace BGP for routing between private networks. Like OSPF, IDRP is a link-state routing protocol, which allows multiple routes to destination, and records link attributes such as throughput, delay, and security characteristics. It also supports authentication of the exchanged route data and supports a hierarchical management of domains called Domain Confederations.

Case Study: Where Is the Data Coming From?

Until you actually take a look at a protocol analyzer, you may not be aware of the importance of knowing exactly how TCP/IP gets data from one place to another. But when you do see all those bytes flowing in and out of your computer, even when (from the user's point of view) the computer isn't actually doing anything, you'll feel much more comfortable knowing which bits are natural and which ones are suspect.

One customer, who manages a medium-size LAN connected to the Internet as well as a number of remote LAN clients that connect over the Internet, called one day after seeing lights on his modem though he wasn't actually using the computer. He observed that even when he wasn't transferring files or browsing the Web, every second or so the modem lights would flash. Instantly his imagination conjured visions of hackers and Trojan horses burrowing into his hard-drive and sending sensitive corporate data out over the network. An examination of the TCP/IP statistics of his computer (he is relatively knowledgeable in networking) showed that no IP packets were going out.

What the manager did not know was that PPP is a true Data Link layer protocol, and some PPP implementations (including his) periodically exchange link data even when no protocol data is being transferred over the link. His mind was set at ease after he closely examined what exactly was being sent and why.

Chapter 4: Sockets and Services from a Security Point of View

Overview

In Chapter 3, you saw the first four layers of the OSI stack—those layers which transfer data, irrespective of what that data is, from one computer to another over the Internet. This chapter focuses on the security implications of the types of data exchanged between computers using these lower-level protocols. We'll talk about common protocols, such as HTTP and FTP, hidden or problematic ones, such as RPC and Finger, as well as obscure ones, like DHCP and SNMP.

A network intruder will look for security weaknesses at every point in your network architecture. If you have adequately locked down the Physical, Data Link, Network, and Transport layers of your network, the wily hacker will simply move up to those protocols and services your network does expose to the Internet. These application-specific protocols are actually much easier to exploit, so many hackers start there and drop down to the Network or Transport level when they need to circumvent a protocol's security mechanisms.

In this chapter, we'll go over each of the most commonly used Internet services, briefly examining each for their weaknesses and abuse potential. First, however, we'll discuss sockets and services in general, identifying typical service vulnerabilities so you can identify potential problems when you need to install services on your own network.

Evaluating Socket-Based Services

Which services are safe to allow through your firewall, which are not safe, and which ones do you do need to keep an eye on? When a new service becomes popular, or when you want to give your network clients a new Internet-based tool, what do you look for when you evaluate the service?

Later in this chapter, we will examine the most common protocols that use IP as a transport and are passed through firewalls. For those protocols, and for any others you may want to support in your network, you should ask the following questions:

- How complex is the service?
- How might the service be abused?
- What information does the service dispense?
- How much of a dialog does the service allow?
- How programmable or configurable is the service?
- What other services does the service rely on?
- What sort of authentication does the service use?

How Complex Is the Service?

Complex services are easier to exploit than simple services. The Echo service, for example, simply transmits back to the client whatever the client sends to it. The Echo service is useful for debugging and for network connectivity testing, but it is difficult to see how the Echo service could be exploited to gain control of the computer running the service. Since the Echo service accepts data from the client, however, it must be programmed to correctly handle being fed too much data at once.

The mail service, on the other hand, is a large, complex piece of software that accepts data (mail) from and returns data to the client, as well as reads and stores data and configuration information on the computer's hard drive. Many mail services (POP and IMAP, for example) require authentication before the client can use the service. SMTP, on the other hand, allows any connecting user to send mail as though it came from any user—even a non-existent one. If the authentication mechanism can be subverted, the passwords hacked, or the service tricked into sending out private data (such as your password file), the hacker can use the service to get enough information to break into your computer through other means, such as FTP or Telnet.

Buffer Overruns and Denial-of-Service Floods

Buffer overruns, as described in Chapter 2, are a potential weakness of any service that receives data from clients. Optimistic programmers expect network clients to play by the rules and only send as much data as would be appropriate for the protocol. For example, the destination address portion of an SMTP message should be somewhere between 4 and 200 characters. But what if the client sends several mega-bytes instead?

A correctly written implementation of the protocol will simply discard the excess data and return an error. An incorrectly written implementation, which does not check how much data is being received, may overwrite its own program code and either crash or (if the sender of the data sent it maliciously) begin executing the excess data as a program.

Early versions of Sendmail had exactly this kind of security hole, and the infamous Internet Worm used this weakness to gain control of target computers and bring the whole Internet to its knees.

A related attack for computers that store data (such as mail) is to keep sending data until the storage space on the server runs out. This is a denial-of-service attack that, although it does not give the hacker control of the computer, renders the computer unusable by its intended clients.

How Might the Service Be Abused?

Some services might be simple and innocuous in themselves, but can be turned to unexpected and detrimental uses. Chargen, for example, is a simple Unix service that sends out ASCII characters over and over. Chargen is a useful network programming and testing tool, because there are certain classes of networking problems that become evident when you can look at a stream of data spanning a whole range of binary representations. A communications channel that clears (or sets) the top two bits of every data word, for example, becomes obvious because the pattern of characters from Chargen will change as well.

An unscrupulous hacker, however, might exploit this protocol by forging a SYN packet (connection request) that redirects the output of Chargen to another computer and port. This way the hacker can flood the target computer with data that doesn't even originate from his own computer!

What Information Does the Service Dispense?

Some services may be simple, terse, and still dangerous to your network security. Finger, for example, was designed to help Unix users contact each other. A Finger request will tell you whether or not there is an account for an individual on a computer, what that account name is, when the user last logged on, additional contact information for the user, and whatever else that user would like to tell the world.

That information is certainly useful if you need to know a coworker's e-mail address or phone extension. It is also incredibly useful for a hacker who wants to verify the existence of an account, find a dormant one, or get clues to the identity and personality of an account holder. You won't find many computers on the modern Internet that support the Finger protocol. The Whois service is another one that you might not want to run on your network due to the amount of information it can give to a network intruder.

Services such as Whois and Finger are excellent timesaving tools for use within an organization, but you should never allow access to these services from beyond your internal network or your intranet.

How Much of a Dialog Does the Service Allow?

A simple service with a regular interface is easier to secure than a complex service that uses an extensive dialogue with the client to provide its functionality. HTTP, for example (disregarding CGI, server applets, and Active Server Pages for the moment), is easy to secure because all the client can do is ask for a resource, and the server does not maintain data about the state of the connection between client requests (i.e. the protocol is stateless).

A stateful protocol is more difficult to secure, especially if the protocol requires client authentication at the beginning of the session and allows for many requests, replies, and state changes after authentication. A stateful protocol must be tested for security holes for every state the client may place the server in. It is possible, for example, to hijack a Telnet session after an authorized client has established the connection and provided correct credentials.

Also, the more dialog a service allows, the more dangerous the service is when compromised. If a hacker arranges the Finger service to run at the wrong permissions level (such as root or Administrator), the hacker would still just get account and contact information from it. An FTP session at the supervisor level, however, could send the hacker any file in the computer. A root Telnet session would allow the intruder to do anything at all, including running programs, shutting down or starting services, replacing operating system code, as well as creating and deleting accounts.

Service Separation

FTP sites, websites, and Telnet hosts perform vital functions in today's Internet-connected networks. They create an Internet presence and facilitate communication within the organization as well as with outside customers and the rest of the world. You can't just decide not to run these services, but you can be careful about how you do it.

Because computers running these services are more vulnerable to being compromised, these services should not be run on computers that also host other internal services (i.e. the file server or a database server). Ideally, each Internet service would be run on its own server, but some services work well together (FTP is often used to update web server content, for example).

Most common Unix utilities were developed before Internet security was a serious problem. New, more secure utilities and services can replace nearly all common Unix utilities. You should seriously consider replacing Telnet with SSH, FTP with SFTP, and POP with SPOP. These utilities are basically the same except that they perform secure authentication and stream encryption rather than sending passwords and data in the clear. If you don't have the option of changing non-secure utilities, use a VPN.

How Programmable or Configurable is the Service?

The more configurable a service, the easier it is to get the configuration wrong. The more programmable the service, the more likely bugs are to pop up, creating loopholes for network intruders to jump through. So, Exchange Server (which has more configuration options than you can shake a stick at) and Internet Information Server (or any other web server that allows you to run arbitrary scripts to generate web pages) are more likely to contain security weaknesses than simpler services, such as the POP service that comes with the NT 4 Resource Kit.

What Sort of Authentication Does the Service Use?

Any service that requires authentication from the client is a potential security risk for two reasons: the authentication protocol may be weak, and users tend to use the same account names and passwords across multiple services.

POP is one example of weak authentication. The username and password are sent from the client to the server unencrypted, and the password is sent complete and unsalted. In POP, the server asks for the username and password, and the client just sends them. Compare this to MAPI (used by Microsoft Exchange), which uses a challenge-and-response protocol. With MAPI, the server requests the username and password, but also sends a value for the client to salt (prepend to) the password before the client hashes (scrambles) the password. The username and hashed password are then sent to the server. The server can compare the hash to a known hashed password to determine if the client should have access to the service. No eavesdropping computer can determine what the password is from the hash, and the same hash cannot be used more than once because the server changes the hash every time.

Another common problem with authentication is insufficiently long passwords. LAN Manager-style passwords, which are used in Windows for Workgroups and Windows 95, are weak because case-sensitivity is ignored in the hash, and because the hash comprises two sections using the same seed that may be cracked in parallel. In addition, older Microsoft networking clients send the same password hashed with both the LAN Manager protocol and the Windows NT challenge-and-response protocol, seriously compromising the Windows NT hash.

Password hashing works by storing an encrypted version of a password rather than storing the password itself. The encryption algorithm is similar to a public-key protocol in that it can only be used to encrypt the password, not to decrypt it. Retrieving a stored password then doesn't reveal the password because the hash cannot be reversed.

Challenge-response protocols are used to ensure that a hacker can't use a password hash. In a challenge-response protocol, a random number is transmitted by the server to the client. The client then encrypts the random number using the password hash and transmits the result back to the server, which uses its copy of the hash to decrypt the random number. If the decrypted random number matches the original random number, then the client has proven that it knows the hash and the server can trust it. On the wire, only a random number and permuted random number have been transmitted, both of which are worthless and cannot be reused.

The purpose of a challenge-response protocol is to keep the hacker from intercepting the passwords as they travel from the client to the server. If the hacker can't intercept the password, he may just try to guess it. This is another area in which many protocols fail. A properly implemented protocol will detect an unusual number (three or greater) of failed password attempts, after which it

will not allow any more logon attempts to that username or from that client. A weak protocol will allow as many attempts as the hacker can perform, and a clever hacker can write a program to perform hundreds of attempts per second, determining the true password by brute force. Windows will by default lock out any account but the Administrator account when there are too many failed password attempts.

Warning Initially, VNC, the popular open-source remote control software, did not lock out intruders and would respond rapidly to attempts. The protocol could be brute-forced in just a few hours. This problem was fixed in 3.3.3r7, so be certain you stay up to date.

It is easier to remember one password than a half-dozen, so many computer users use exactly the same password for all of their computer accounts. This means that if a network intruder penetrates one computer and captures the password list for that computer, one or more passwords and usernames from that computer are very likely to also work in another computer that is otherwise secure. Your password-protected service gives the hacker a double resource—if the hacker can find a password that works on that service, she'll try it elsewhere, and if she can find a password elsewhere, she'll try it on that service as well.

One Key Fits All

For convenience's sake, some services use the operating system username and password instead of recording a separate username and password pair for the service. While this feature makes life easier for the user, it also makes life easier for the hacker. When the hacker has obtained access to the service, it means that he has also obtained access to the underlying operating system. Internet Information Server (for password-protected web pages) works this way, as do the POP and SMTP services that can be installed with it.

Your Network Profile

One thing that you should realize as you consider which services you will expose to the Internet is that the more services you choose to expose, the less secure your network will be. Each protocol you decide to allow may have a low probability of being compromised, but those probabilities are additive and it only takes one hole to negate all your security efforts. So remember, less is more—in this case, fewer services exposed is more security for your network.

In addition, hackers use the services you expose to profile your servers and they use that information to select which attacks to run in an attempt to penetrate your network. If you expose all of the ports that a default Windows server exposes, then the hacker is much more likely to pull out his Windows hacking scripts rather than his Linux hacking scripts, for example.

DNS, The Essential Service

One of the most important higher-level services in the Internet Protocol Suite is the Domain Name Service, and it deserves a bit more explanation than the rest of the protocols because it is so intimately tied with the operation of the Internet. The lower-level Internet protocols all use IP addresses to identify source and destination computers, but people have a hard time remembering numbers. So, the architects of the Internet developed a service that would translate human-friendly Internet addresses like <http://www.microsoft.com/> and <http://www.linux.org/> into their corresponding computer-friendly IP addresses.

Dot Com and All That

In the beginning, people just kept a list of the human-friendly Internet addresses and their corresponding IP addresses (the host file is a remnant of this), but as the Internet grew these files got unwieldy and a more flexible method was needed. The Internet architects came up with a new scheme that allowed an organization (or a person) to request an Internet address from a central authority and then expand on that name using their own authority. Thus was born the Internet Domain Name Registration system.

Human-readable Internet addresses, like IP addresses, contain dots. But Internet addresses can have as few as one dot or many more than four (although it is a rare address that will have more than a half-dozen). The dots function as separators, and the last portion of the address is called the top-level domain. Common top-level domains include .com, .org, .net, .mil, .gov, .edu, and two-character country codes such as .uk, .us, .cz, .au, and .oz.

When you request a domain name such as `memetech.com`, `litvault.org`, or `robinsoncrusoefamily.org`, you are staking out space on the Internet in a similar manner to when you obtain a range of IP addresses. It is up to you to further subdivide the name space. When you request the name, you provide the top-level service with the IP address of two servers that will handle name requests from your name space. When a computer wants to contact `www.memetech.com`, for example, it is up to the domain name servers that you specify to perform the translation from `www.memetech.com` to the actual IP address. Your servers can put any computer name you want before `memetech.com`, including `this.is.a.very.long.domain.name.memetech.com`.

A large organization or a service may further subdivide the name space. BigCorp, for example, might have separate domain name servers for each of its divisions in Asia, Europe, the US, and Africa. In this scenario, one domain name server would handle requests such as `http://www.asia.bigcorp.com/` and a different one would handle requests such as `http://www.us.bigcorp.com/`, allowing each division to configure their networks independently but maintain overall BigCorp control of the `bigcorp.com` domain.

DNS Queries

When you type `http://www.microsoft.com/` into your web browser, your computer needs to know what IP address corresponds to that Internet address. Microsoft's DNS (Domain Name System) server is the computer that contains that information. How does that information get from the Microsoft DNS server into your computer?

Your computer has the IP addresses of one or two DNS servers that it can ask about domain names (and unless you work in Redmond, neither one is likely to be the Microsoft DNS server that has the resolution you need). Your DNS servers most likely contain only Internet address-to-IP address mappings for your local network and for a few frequently accessed external locations. When your local DNS server cannot satisfy a DNS mapping request, it will either redirect your computer to another DNS server or query that DNS server itself and pass the results back to your computer.

If the mapping query is for an address in a domain foreign to the DNS server, the DNS server redirects the query up the chain of servers toward (or to) a root server (such as the server for the .com, .edu, or .gov domains). A server that has recorded and delegated name service for a domain will redirect the query down to the DNS server that has responsibility for the domain. See Figure 4.1 for an example.

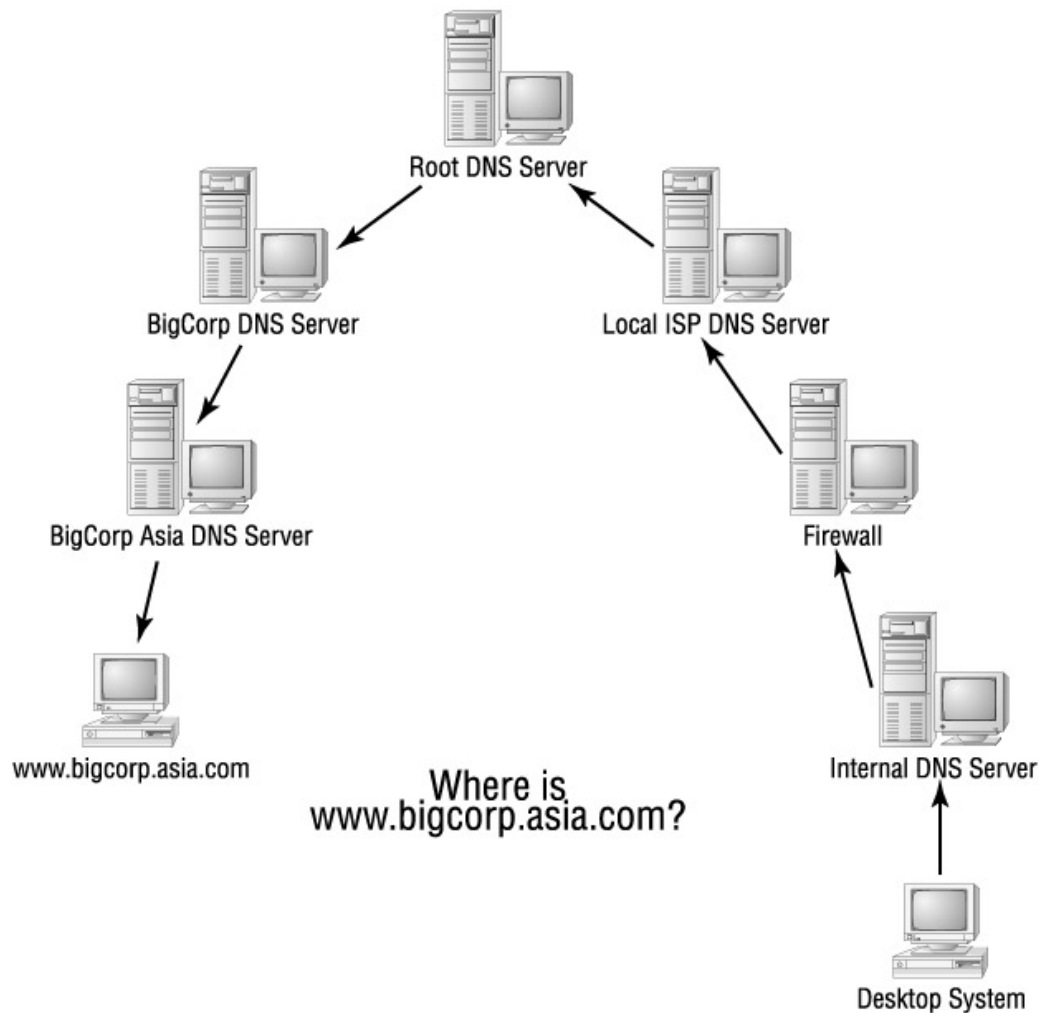


Figure 4.1: The DNS system distributes control of Internet domain names.

DNS Messages

Your computer makes a DNS request by sending a UDP packet in the DNS message format to port 53 of the DNS server. DNS uses UDP rather than TCP because it has one short message that will fit in a UDP packet to send to the server, and the connection establishment overhead of TCP is not necessary for DNS. The message has a fixed 12-byte header and 4 variable-length fields, as shown in Figure 4.2.

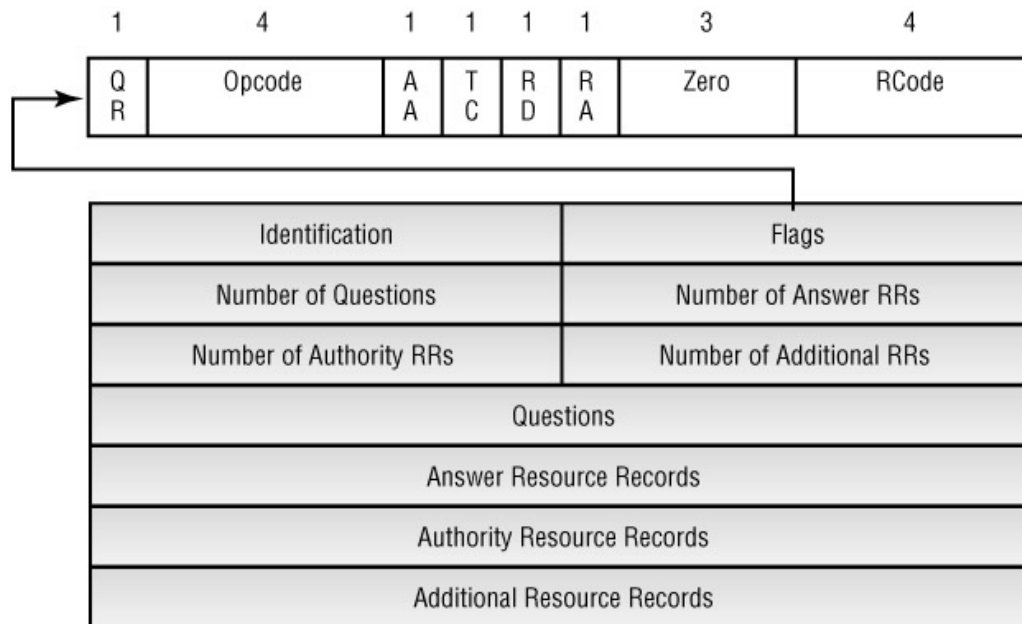


Figure 4.2: A single DNS message may contain multiple address resolution queries or responses in the Question RR and Answer RR fields of the DNS UDP packet.

The DNS message fields are as follows:

- **Identification** This is set by the client, so that it can identify the DNS server response.
- **Flags** This field classifies the message as follows (See Figure 4.2):
 - ◆ **QR** 0 for query or 1 for response.
 - ◆ **Opcode** 0 for Standard Query, 1 for Inverse Query, or 2 for Server Status Request
 - ◆ **AA** Authoritative Answer (True/False)
 - ◆ **TC** Truncated (True/False)
 - ◆ **RD** Recursion Desired (True/False)
 - ◆ **RA** Recursion Available (True/False)
 - ◆ **Zero** These three bits are zero.
 - ◆ **Rcode** 0 for No Error, or 3 for Name Error
- **Number of Questions** 1 or more for query, 0 for reply
- **Number of Answer RRs** 0 for query, 1 or more for reply
- **Number of Authority RRs** 0 for query, 1 or more for reply
- **Number of Additional RRs** 0 for query, 1 or more for reply
- **Questions** Each question is composed of a string containing the Internet address being queried, a query type, and a query class (which is 1 for Internet addresses). Common types are as follows:
 - ◆
 - 1 A The IP Address
 - 2 NS The Name Server for the domain
 - 5 CNAME The Canonical Name (if a computer responds to more than one name, this is the main one)
 - 12 PTR The Pointer Record (looks up the name corresponding to an IP address)
 - 13 HINFO Host Information (computer details)
 - 15 MX The Mail Exchange Record (which host handles mail for this Internet Address)

252 AXFR The Request for Zone Transfer

255 ANY The Request for all records

- **Answers, Authority, and Additional Information** All share the same Resource Record format, as follows:

- ♦ **Domain Name** This is the Internet address that information was requested for.
- ♦ **Type** This is the same as the Type field in the query.
- ♦ **Class** This is usually set for 1 for Internet addresses.
- ♦ **Time-To-Live** This is the number of seconds that the data may be cached by the client (typically two days).
- ♦ **Resource Data Length** This is the amount of space taken by the resource data.
- ♦ **Resource Data** This is the four-byte IP address for a type 1 (a record) response.

So when you type `http://www.microsoft.com/` into your web browser, your computer fires off a UDP packet to your DNS server asking where Microsoft is. The DNS server either sends back a UDP packet containing that information, or it redirects your computer to another DNS server that might have it instead. Eventually, your computer gets back an Answer record containing the IP address, and the web browser can then use it to open a TCP socket to Microsoft's web server.

Reverse Name Lookup

Sometimes you already have the IP address to a computer, and you want to know what the Internet address for that computer is. Web server administrators, for example, often need to know which domains most of their traffic is coming from. More importantly, if you suspect that a hacker is breaking into your system, you need to be able to translate the hacker's IP address into a domain so that you can contact the administrator of the network that the hacker is coming from. How do you look up an IP address?

Note The different kinds of Internet addresses can be confusing to people who are just learning how the Internet protocols work. Just remember—IP addresses are for computers and are made up of four numbers separated by dots. Internet addresses (also called Internet names) on the other hand, are meant to make sense to humans, and have the familiar endings such as `.com`, `.edu`, `.mil`, `.net`, or `.org`.

When you are allocated a block of IP addresses and you request a domain name to go with it, you are also given control of a portion of the DNS name space that corresponds to the IP addresses you have control over. The top-level domain name for this portion of DNS is not `.com` or `.org`, but rather `.in-addr.arpa`, which (like the rest of the DNS system, and unlike IP addresses) gets more specific as values are added to the left of it.

If you want to look up the address `128.110.121.144`, you would request the PTR record for `144.121.110.128.in-addr.arpa`, and you would receive back the actual Internet address for that computer, if there is one registered for it. This is why your own DNS server must be configured for that weird numerical `in-addr.arpa` domain in it as well as the domain you requested.

Reverse Lookup

Many services will not allow a network client to connect to it if the client does not have an Internet name as well as an IP address configured for it. While a computer without an Internet name is not proof-positive of hacking, it is indicative of a domain with lax administration. If the protocols you support allow it, and your network can handle the overhead of looking up every IP connection attempt, lock out these nameless computers.

Coordinating Name Servers (and Subverting Them)

When you obtain a domain name (or more than one), you are required to provide the IP addresses of at least two DNS servers that will handle DNS requests for your domain(s). Why two? So that if one fails, the other can take over its responsibilities. One is designated the primary DNS server and has several secondary servers to backup its data.

Primary and secondary DNS servers are typically configured to stay in sync through the exchange of zone transfer messages. The primary can preemptively send the zone transfer message, or the secondary can request that the primary send one. Zone transfers can be incremental or complete.

DNS servers make juicy targets for hackers because the hacker can use a compromised name server to redirect unwitting clients to her own computer. As a result, impersonation and man-in-the-middle attacks are easy to perform. If your DNS servers aren't particular about where zone transfer packets come from, the hacker can feed false information to the secondary server and then perform a denial-of-service attack on the primary (perhaps through a ping-of-death or smurf attack). The hacker is then in charge of your network name space, not you.

Note With a very small network you can disable zone transfers and simply update the tables of your internal DNS servers by hand. For larger networks this becomes impractical, and you must either have firewall software that discriminates between regular DNS queries and zone transfers, configure your DNS server to allow zone transfers only for your secondary servers, or you must have DNS proxy software that will allow only safe domain name requests.

Common Internet Services

This section profiles the "hackability" of various common Internet services. Internet services at the Application layer are themselves not layered; they do not have a unified authentication service, a uniform encryption service, or any other dependence upon other protocols with the exception of DNS. This means that each of these services has implemented its own security procedures—some well, some very naively.

The following services (and their Port and Protocol types) are covered in this section:

- DNS (53 UDP)
- FTP (20 and 21 TCP)
- HTTP (80 TCP)
- IMAP (143 TCP)
- NTP (123 UDP)
- POP (110 TCP)
- SMTP (25 TCP)
- SSL (449 TCP)

DNS (53 UDP)

While this protocol is a prime target for network intruders, you can't disable it without disabling all your network clients as well. You should protect your DNS servers (as described later) by blocking zone transfer packets or by using a DNS proxy service.

The following bulleted items summarize this protocol's security profile (we'll provide a similar

bulleted list for each protocol we discuss):

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—Minimal
- Dialog—Minimal
- Programmability and Configurability—High

File Transfer Protocol (20 and 21 TCP)

FTP is a useful command-line protocol for transferring files over the Internet. FTP is often used to remotely update web content on HTTP servers. For this reason, among others, it may be necessary to allow FTP traffic through your firewall.

FTP's development predates the development of firewalls and it is therefore a little more difficult to pass through a firewall than newer protocols such as HTTP. When a client opens a connection with the service (using port 20, the command channel), the server opens a second connection to the client (using port 21, the data channel). If the firewall is using IP translation to hide the client computers behind a single public IP address, the data channel connection attempt will fail unless special measures are taken in the firewall to identify and pass through the incoming data channel.

You should be exceedingly careful in configuring FTP security because FTP establishes a dialog with the client in which the client can browse files on the FTP server and download them, and because FTP authentication is made using operating system usernames and passwords. Even if someone accesses the FTP server as the "anonymous user," the user can gain access to critical operating system files if you have set up file and directory security incorrectly, (especially if you have established symbolic links that allow the anonymous user out of the typical safety sandbox).

When you set up an FTP server for access that is external to your network, do not use the same account names and passwords on the FTP server as are used for LAN logon.

Here's a rundown of the security characteristics of FTP:

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—Medium
- Dialog—Complex
- Programmability and Configurability—High

HTTP (80 TCP)

The Web uses the HTTP protocol to transfer text, video, sound, and even programs over the Internet. Initially, web servers were very simple (merely sending out to a client whatever page the client requested), but the exploding World Wide Web demands more and more features from web servers. Now a web server is a complex piece of software, with many configuration options, a complicated dialog, and infinite programmability.

The hacker exploitation of HTTP can go both ways—a hacker may try to exploit your website using HTTP, and a hacker website may contain dangerous web page components such as malicious ActiveX controls or Java applets.

ActiveX controls are an Internet Explorer feature. An ActiveX control, once downloaded and run on

a client computer on your network, can do absolutely anything any other program on that computer can do. You should require that on your network only those ActiveX controls that have been digitally signed by organizations you trust will be downloaded. You can use the Internet Explorer Administration Kit to lock down this Internet Explorer setting. If you can get away with it, disable ActiveX controls entirely.

Java is a little safer. Make sure that all of the computers in your network are configured not to allow Java applets access to hardware resources unless they are digitally signed by organizations you trust.

On the server side, be extremely careful with remote web administration software. Most of the website hacking done by Internet vandals has been accomplished by exploiting security holes in remote website management tools.

Scrutinize server-side applets and CGI scripts. Do not make script directories browsable. Do not allow arbitrary scripts to be uploaded. Do not allow scripts to be executed from directories that contain other web data.

If you can, maintain web page usernames and passwords separately from operating system usernames and passwords. Log web access, and look for unusual patterns (excessive 404 errors, etc.).

Security characteristics of HTTP:

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—Medium
- Dialog—High
- Programmability and Configurability—High

IMAP (143 TCP)

This is the protocol used by network clients to retrieve mail from servers that are configured to retain e-mail on the server rather than transfer it to the client. The protocol itself, while more complex than POP, is slightly more secure (passwords aren't sent in the clear, at least).

Security characteristics of IMAP:

- Complexity—Simple
- Abuse Potential—Medium
- Information Sensitivity—Medium
- Dialog—Low
- Programmability and Configurability—Low

NTP (123 UDP)

This is the protocol used by network devices, including firewalls, to reliably update and synchronize their time and date settings. Many implementations of NTP have proven vulnerable to buffer overrun exploit that allows remote root access. This is the perfect example of why even the simplest services should be inspected for security regularly.

Security characteristics of NTP:

- Complexity—Simple
- Abuse Potential—Medium
- Information Sensitivity—Medium
- Dialog—Low
- Programmability and Configurability—Low

POP3 (110 TCP)

The Post Office Protocol allows clients to check their e-mail over the LAN or over the Internet. POP is easy to configure and use, but the protocol is a little too simple—it doesn't encrypt usernames or passwords. Avoid allowing access to internal mail accounts from outside the firewall using POP—and if you do, do not allow POP account names and passwords to be the same as LAN usernames and passwords. Consider using IMAP instead.

Security characteristics of the POP3 protocol:

- Complexity—Simple
- Abuse Potential—Medium
- Information Sensitivity—Medium
- Dialog—Minimal
- Programmability and Configurability—Low

SMTP (25 TCP)

Most of the mail exchanged over the Internet is done using the Simple Mail Transport Protocol (SMTP). This protocol simply accepts mail in a simple dialog (without checking the authority or even the identity of the sender).

Although the protocol is simple, the software that processes the mail (once it's received) is often not so simple. Many SMTP packages have complex configuration options and forwarding rules, and, if incorrectly configured, these can adversely affect network performance or crash the mail server when large amounts of mail are being processed. Also, the lack of sender authorization leaves SMTP open to spam attacks and e-mail flooding.

Unfortunately, if you want to receive Internet mail you need to support SMTP. You should choose mail server software that is as bulletproof as possible and use care when configuring it, paying attention to details like available hard disk space, network bandwidth, and so on. Install a server-based virus scanner to sanitize e-mail attachments as well.

Security characteristics of the SMTP protocol:

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—Medium
- Dialog—Minimal
- Programmability and Configurability—High

HTTPS (449 TCP)

HTTP transfers (on port 80) traverse the Internet unencrypted, so web traffic using it should not be trusted. Internet commerce requires trust, however, as do private communications and secure Intranet use. Fortunately, the Secure Socket Layer provides a way for HTTP traffic to be encrypted

between the client and the server. Other protocols may use SSL as well to encrypt their session data, but the network clients and servers must be written to allow this option. SSH is one other such service, as is SFTP (FTP over SSL).

Security characteristics of HTTPS:

- Complexity—Simple
- Abuse Potential—Low
- Information Sensitivity—Medium
- Dialog—Minimal
- Programmability and Configurability—Low

Other Common Services

DNS is one service you must allow through your firewall in one manner or another, because without it your network clients won't be able to find anything. There are many more services you may want to support on your network, or that you may elect to block depending on the needs of your network users. Each has its strengths and vulnerabilities, which we'll examine using the criteria established at the start of this chapter. The protocols we'll discuss can be divided into three categories, as follows. Note that nearly all services are available for all platforms through the use of third-party applications. We've divided the services based on the standard configurations of these operating systems.

Windows-Specific Services:

- NetBIOS (137, 138, and 139 TCP)
- RPC Locator (Windows) (135)
- Terminal Services (3389 TCP)

Standard Unix Services:

- Chargen (19 UDP and TCP)
- Daytime (13 UDP)
- Discard (9 UDP and TCP)
- Echo (7 UDP)
- Finger (79 TCP)
- NFS (2049 TCP and UDP)
- Quote (17 UDP)
- RPC (Unix) (111 UDP)
- RSH (514 TCP)
- SSH (22 TCP)

Platform Neutral Services:

- Telnet (23 TCP)
- TFTP (69 UDP)
- BootP/DHCP (67 and 68 UDP)
- LDAP (389 TCP and UDP)
- SNMP (161 UDP)
- VNC (5800+, 5900+ TCP)

Windows-Specific Services

There are a number of services that are specific to Windows networking. If a hacker sees these ports open on your firewall, then he can be sure that there are Windows machines inside to be compromised. NetBIOS and RPC Locator in particular are easy to exploit and Terminal Services is an attractive target. So, if you need to access these Windows-specific ports from outside your network, you should consider blocking them at the firewall and requiring the remote client or network to establish a VPN connection to your firewall before using these ports.

NetBIOS (137, 138, and 139 TCP)

NetBIOS is the protocol used by Microsoft Windows networking to connect LAN clients to file and print servers. NetBIOS will run over IPX, NetBEUI, and TCP.

You should definitely not allow NetBIOS traffic to pass your firewall in either direction. NetBIOS is easily hacked, and many exploits exist on hacker websites. If you want to link your LANs over the Internet, you should use an encrypted IP tunnel to convey the NetBIOS IP packets through the Internet.

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—Low

RPC Locator (135 TCP)

RPC (Remote Procedure Call) Locator is the protocol used by Microsoft Windows networking to allow programs to find network services that use the RPC protocol. RPC Locator will run over IPX, NetBEUI, and TCP.

You should definitely not allow RPC Locator traffic to pass your firewall in either direction. RPC Locator is easily confused and Windows can crash if it is fed bad data.

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—Low

Terminal Services (3389 TCP)

Terminal Services gives the attaching computer complete remote control of the machine. The authentication is encrypted and the session data may be encrypted if you choose, but allowing indiscriminate access to this service from the Internet—at-large is incredibly risky. If you require remote control of servers within your network from the outside, then Terminal Services is best used in combination with another security mechanism, such as a VPN.

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High

- Programmability and Configurability—Low

Standard Unix Services

Unix has a bevy of simple legacy services from early in the development of the Internet—when security was less of an issue and it was useful to have a service that cycled through the character set, so you could see if one of the routers between your computer and that host was chopping off the top bit of all the bytes in the network packet and thereby garbling your e-mail. Since then, the world has standardized on eight-bit bytes and hackers have learned to exploit unchecked buffer copies; so while the majority of these protocols are harmless, it makes no sense to leave them exposed at the firewall.

Chargen (19 UDP and TCP)

Chargen continuously sends out the printable ASCII characters. It is useful for testing network applications. Any service that could be stopped or swamped by a stream of ASCII characters is broken anyway and shouldn't be let past your firewall. It is extremely unlikely that a network intruder could use Chargen to break into your system.

- Complexity—Simple
- Abuse Potential—Chargen can be redirected to flood other unsuspecting computers.
- Information Sensitivity—None
- Dialog—None
- Programmability and Configurability—None

Daytime (13 UDP)

This service sends the date and time at the server to the client. It would take a very clever hacker to find a security weakness in this protocol.

- Complexity—Simple
- Abuse Potential—Minimal
- Information Sensitivity—None
- Dialog—Minimal
- Programmability and Configurability—None

Discard (9 UDP and TCP)

This protocol throws away any data sent to it. It is useful for developing network tools. It is as secure as it is useless.

- Complexity—Simple
- Abuse Potential—Minimal
- Information Sensitivity—None
- Dialog—Minimal
- Programmability and Configurability—None

Echo (7 UDP)

Echo repeats to the connected client whatever the connected client sends to it. It is useful for testing network applications. It is extremely unlikely that a network intruder could use Echo to break into your system, as long as Echo properly manages its input buffers.

- Complexity—Simple
- Abuse Potential—Minimal
- Information Sensitivity—None
- Dialog—Minimal
- Programmability and Configurability—None

Finger (79 TCP)

The Finger service was designed to help network users communicate by providing system information, such as the last time a user checked their e-mail, and real-world data, such as the user's office hours, telephone number, or current projects. Unfortunately, this data is as good as gold to hackers looking for potential account names and passwords. Also, some hackers will even go so far as to call an office pretending to be the help desk staff and trick users into giving up their account names and passwords over the phone.

- Complexity—Simple
- Abuse Potential—Average
- Information Sensitivity—High
- Dialog—Minimal
- Programmability and Configurability—Average

NFS (2049 TCP and UDP)

NFS is the Unix equivalent of NetBIOS; it gives LAN clients access to file server storage. If you need to allow remote clients access to NFS resources, establish an encrypted tunnel to do it, don't just open up the NFS ports. Improperly configured NFS servers can be easily exploited to gain root access, and it's easy to spoof NFS legitimate connections.

- Complexity—Complex
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—Medium

Quote (17 UDP)

This protocol merely sends any connecting client a random selection from a file full of quotes. Quote provides little leverage for abuse.

- Complexity—Simple
- Abuse Potential—Minimal
- Information Sensitivity—None
- Dialog—Minimal
- Programmability and Configurability—None

RPC (Unix) (111 UDP)

Remote Procedure Call is a protocol that allows two computers to coordinate executing software. A program on one computer can use RPC to transfer the execution of a subroutine to another computer, and have the result returned via RPC to the first.

RPC is a fragile service, and most operating systems cannot handle arbitrary data being sent to an

RPC port. RPC is best used in trusted LAN environments, and you should not let RPC traffic through your firewall.

- Complexity—Medium
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—High

RSH (514 TCP)

The Remote Shell protocol makes up for deficiencies in Telnet. There are always dangers when you allow remote command-line access to computers through your firewall, and RSH is better than Telnet because RSH at least protects the passwords as they are exchanged between the client and the server, but SSH is better than either because it encrypts the session data as well. Use SSH for remote console access and block Telnet and RSH.

- Complexity—Medium
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—Low

SSH (22 TCP)

The Secure Shell protocol makes up for deficiencies in RSH and Telnet. SSH uses SSL encryption to securely authenticate communications and encrypt session data. You can configure it to allow usernames and pass-phrases or cryptographic key exchange (or both) to authenticate the user, public keys to specify authorized client computers, and a range of algorithms to encrypt the session data with. If you are going to allow any kind of remote control through your firewall this is a good choice.

- Complexity—Medium
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—High
- Programmability and Configurability—Medium

Platform Neutral Services

There are some services that you may find running on both Windows computers and Unix computers because their universal utility. Not all of them must be exposed to the Internet, however, because many of them are primarily used in a LAN environment; DHCP is one such example. Others—Gopher, for example—have been superceded by later protocols. Some others are current and often used in an Internet setting but must be treated with care; LDAP is one such.

Telnet (23 TCP)

Telnet is extremely simple—it's just a connection opened to a command line interpreter. Whatever you type is sent to the interpreter on the server, and the interpreter's response is returned to you. The data traffic is not encrypted, and when you log on, the username and password are readable by any computer on any intermediate LAN.

Do not allow Telnet access to computers inside your firewall. If you require command-line access, use a more secure protocol such as SSH.

- Complexity—Simple
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—Minimal
- Programmability and Configurability—None

TFTP (69 UDP)

TFTP is used with BootP and DHCP to allow diskless workstations to load their operating system and other configuration over the LAN. TFTP does not have the two-channel problem that FTP has (and therefore it interoperates well with a firewall), but there is little reason to allow TFTP through a firewall when you already have FTP and HTTP for file distribution. Also, hackers have developed tools for using unprotected TFTP servers as pirated software dumping grounds, so you should use a more secure file transfer protocol through your firewall.

- Complexity—Simple
- Abuse Potential—High
- Information Sensitivity—Medium
- Dialog—Minimal
- Programmability and Configurability—Low

BootP/DHCP (67 and 68 UDP)

BootP was developed as a simple mechanism for allowing simple network terminals to load their operating system from a server over the LAN. Over time it has expanded to provide for centralized control of many aspects of a computer's identity and behavior on the network, including allocating IP addresses, configuring gateway, DNS, and router settings, dispensing NetBIOS names, as well as downloading operating system files. The greatest danger from BootP and DHCP is from a network intruder impersonating a DHCP server on your network and thereby misconfiguring the DHCP clients. As long as you do not allow DHCP to pass your firewall, you should be able to use DHCP internally without problems.

- Complexity—BootP is simple; DHCP is a little more complex.
- Abuse Potential—Medium
- Information Sensitivity—Medium
- Dialog—Minimal
- Programmability and Configurability—Medium

LDAP (389 TCP and UDP)

The Lightweight Directory Access Protocol is a flexible and distributed way of maintaining contact information (including usernames and passwords) over the Internet. Several Internet services use LDAP to maintain user information rather than relying on the operating system user accounts. This is more secure because it separates operating system functionality from service functionality, and a hacker who gets a service password will not necessarily be able to log on to the server with it. If you want to maintain contact information on your network to facilitate communication with people in your organization, you should consider using LDAP instead of Finger and Whois.

- Complexity—Complex

- Abuse Potential—Medium
- Information Sensitivity—Medium
- Dialog—Medium
- Programmability and Configurability—Medium

SNMP (161 UDP)

The Simple Network Management Protocol is a useful tool for remotely managing network devices such as routers, servers, hubs, clients, and terminal servers. You can use it to enable and disable ports, measure bandwidth, reboot devices, and gather statistics. However, it should be used to manage your network only, not to allow hackers to watch every aspect of the data flow on your network. Block SNMP traffic at your firewall.

- Complexity—Medium
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—Minimal
- Programmability and Configurability—Medium

VNC (5800, 5900 VNCTCP)

Virtual Network Computing (VNC) allows for remote control of computers' desktops, much like Terminal Services does for Windows. Unlike Terminal Services however, VNC is cross-platform—clients exist for Windows, Unix, and many other operating systems besides. While the authentication between the VNC server and the client is encrypted the session information (keystrokes and window contents exchanged) are not. If you require remote control of computers inside your network from the outside, consider requiring other security mechanisms as well—such as a VPN tunnel between the client and the server network.

- Complexity—Medium
- Abuse Potential—High
- Information Sensitivity—High
- Dialog—Medium
- Programmability and Configurability—Low

Case Study: Leaving the Barn Door Open

One of the benefits of being a consultant who actually fixes things and makes life easier for client companies is that, when an employee leaves for greener pastures, they often will take you with them to fix problems in those new, greener pastures. One day, a former employee of a customer of mine called up and asked if I could evaluate the security of his new employer's network. I said that of course I could—and that for this excellent customer I would perform an initial, over-the-Internet evaluation for free. I agreed that I would present the over-the-Internet results to the customer and, at that point, discuss the cost of a full security evaluation of their network, including hardware, software, internal, and external practices.

I then gathered a few widely available Internet tools and began my remote evaluation. The first thing I discovered was that their web server computer (accessible through their firewall) had ports 137 through 139 open. Not only did this identify the computer as a Windows NT machine, it showed that the NetBIOS service was accessible over the Internet, protected only by Microsoft LAN authentication. A quick check showed that the administrative account was still Administrator, and I crafted a small program to repeatedly try passwords (distilled from common password lists I

generated or grabbed from Internet websites). I then got to work looking for other ways into the system.

I didn't get very far, though, because an hour-and-a-half later the password testing program came up with the Administrator password and allowed me in. The password was short and simple and all lowercase. I immediately phoned our contact, let him know that he did indeed have at least two problems, and set up a time to come in and do an exhaustive evaluation.

Chapter 5: Encryption

Overview

For hundreds of years people have used codes and ciphers to protect important information from the view of others. Computers store, manipulate, and transmit information swiftly and accurately. Codes and ciphers that were difficult and time-consuming to perform by hand now take only milliseconds on a computer, so encryption has found a new home in computer operating systems and networks.

Firewalls use codes and ciphers for two vitally important purposes:

- To prove the identity of a user (authentication)
- To hide the contents of a data stream (encryption)

This chapter will first examine what encryption is and how it has developed from a tool for spies and diplomats to become an integral part of modern communications. Next, you'll see how computer networks use encryption to secure your files, keep out unauthorized users, provide a secure channel for communications, and identify trusted computers or users. Finally, you'll be introduced to the several kinds of encryption and the strengths of and uses for each. In addition, you'll be given a glimpse of how a network intruder might attempt to circumvent encryption measures on your network.

This chapter covers encryption in general. Encryption as it is used in firewalls specifically for Virtual Private Networking is covered in Chapter 9.

How to Keep a Secret

The primary purpose of encryption is to keep secrets. Encryption was first used to protect messages so that only the person knowing the "trick" to decoding the message (or the *key* in the jargon of cryptographers) could read the message. History is full of clever codes and ciphers used by kings and princes, spies, generals, business people, and inventors. As you read through this chapter, you'll see examples of historical codes and ciphers, and learn how the concepts illustrated by those examples are used in computers today.

A *cipher* protects a message by rearranging it or performing modifications to the encoding, rather than the meaning, of the message. A *code* is an agreed-upon way of keeping a secret between two or more individuals. A *key* is a bit of information that is required to decrypt a message, usually in the form of a value that is used with a cipher to encrypt a message. The key must be kept secret in order for the message to remain private.

Ciphers

One example of a cipher is used by children around the world: Pig Latin (no offense is intended towards pigs or aficionados of Latin). The secret is simple—the first consonant is moved to the end of the word and the "ay" sound is appended to it. If the word begins with a vowel, simply append "ay." Thus: "Isthay isay anay iphercay orfay implesay essagesmay."

Many dialects of the Pig Latin cipher exist; you could use the keys "oi" or "ah" instead. What makes this cipher different from a code like, say, the backwards day code (wherein words take on their

opposite meanings on the agreed upon "backwards day") is that you can encode any message in pig Latin. This ability to encode any message (or plaintext, as unencrypted messages are called), regardless of subject, is a common feature of all ciphers. This is not a feature of codebooks. Codebooks operate by obscuring the meaning of words in messages and are limited to the words in the book, whereas ciphers obscure the words being used regardless of what the words actually are. Backwards day is a code; Pig Latin is a cipher.

A cipher almost as simple as Pig Latin that is being used on the Internet and on computer bulletin boards around the world today is the Rot(13) cipher. If you've seen an e-mail or newsgroup message that looks like a jumble of random letters and has <rot13> at the top, then you have seen a message using this cipher. The trick to decoding it (the trick is called the *algorithm* in the jargon of cryptographers) is to rotate each letter through the alphabet 13 places. For example, A would be replaced with N, B would be replaced with O, W would be replaced with J, and M would be replaced with Z. Vg'f abg n irel pbzcyvpngrq pvcure. This is a simple *substitution cipher*, so called because each letter of the alphabet is replaced with a substitute letter. The key is to know which letters are replaced with which other letters—in this case, the letter 13 letters away from it (it doesn't matter if you rotate backward or forward, because there are 26 letters in the English alphabet).

The Rot(n) cipher is a generalization of Rot(13), in which you can select the number (n) of rotation, rather than just automatically using 13. Rot(n) has a key input, like DES and other private key ciphers that you can specify to use for encryption and decryption. If you want to use Rot(n) to encode or decode a Rot(13) text, you would use 13 as the key. Figure 5.1 shows how the Rot(n) substitution cipher works.

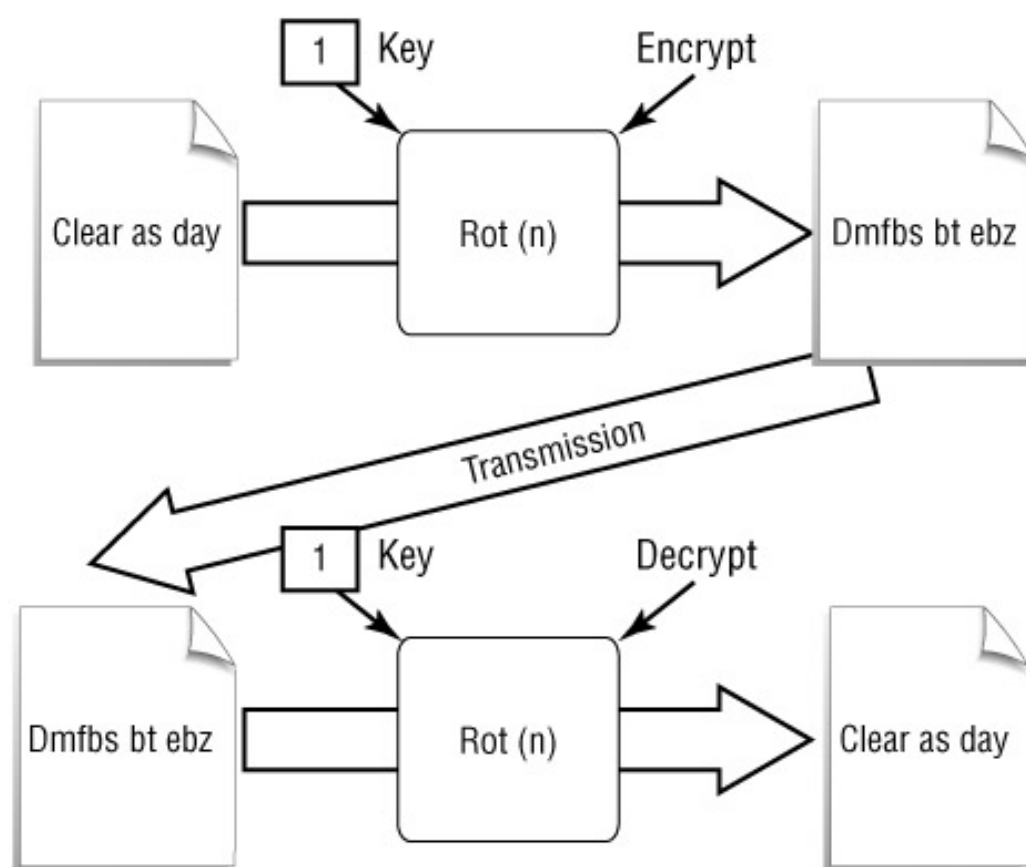


Figure 5.1: In the Rot(n) substitution cipher, the plaintext is rotated several characters for encryption into ciphertext, and then rotated several characters back into plaintext for decryption.

Rot(n), like Pig Latin and backwards day, is not very hard to figure out. Any message encoded with a simple substitution cipher (if the message is long enough—25 characters is enough for most

messages) can be decoded simply by counting the frequency of each letter. The letter E, for example, is the most common English letter and will probably be the one at the top of the frequency list. Common two- and three-letter combinations of letters are also clues to the hidden identity of substituted letters.

Another kind of cipher is the *transposition cipher*. Instead of replacing the letters in a message, this kind of cipher rearranges them. Imagine writing the letters of a message in the squares of a piece of graph paper. You would then copy the message to another sheet of paper, but instead of copying the letters from left to right you would do so from top to bottom. To decipher the message, you would put the letters back on graph paper from top to bottom and then read them from right to left, as usual.

A weakness of ciphers is that if someone knows the cipher you are using and the key you have chosen (say you are using the cipher PigLatin97 with the secret key "chu"), then everything you send or have sent with that cipher and key can be read by that person. This makes it very important to choose a cipher that is difficult to crack and a key that is difficult to guess.

Keeping Secrets Automatically

Encryption and decryption take a long time to perform by hand, and when your computing tools are limited to a pencil and paper (and, of course, the substitution alphabets or codebook) you can only use the simplest of ciphers. However, for hundreds of years the manual process was the only way, until the advent of the Industrial Revolution and the invention of calculating machines.

The various national governments of the 1920s developed rotor machines to automate the encryption and decryption process of cryptographic substitution. The ability to send secure and lengthy messages from the headquarters of the various armed forces to remote locations over a medium that anyone could eavesdrop on (the newly invented radio) proved to be a crucial aspect of modern warfare. Secure communications often meant victory, and broken codes often meant defeat. The most popular encryption machines had rotors and were primarily mechanical.

Each machine contained a number of rotors, and each rotor was wired to replace one letter with another letter. The rotors would rotate at different rates, so a machine with rotors that had 26 positions (one for each letter of the English alphabet) would have a period of 26 raised to the number of rotors. This means that a rotor machine with three rotors would encode the same letter with the same encrypted value every 20,000 or so letters of the message. The Enigma machine used by the Germans in the late '30s was the most famous rotor machine of the time and was fiendishly difficult for British cryptanalysts to crack.

Keeping Secrets Electronically

The development of electronics and computers gave code makers and code breakers a whole new arsenal to work with. They were able at last to develop ciphers that were much too complex to perform by hand. They could also program computers to automatically try many different combinations of keys and ciphers much more quickly than human beings were able to. Many ciphers were developed in secret, and only governments or large corporations had the computing power necessary to use or break the codes.

One algorithm that was developed in secret but then released for use by the public as well as the government (but only for "Unclassified but Sensitive" information) is the *Data Encryption Standard*, or *DES*. It is a symmetric algorithm, which means the same key is used for encryption and decryption, and uses a 56-bit key. DES is widely used in commercial software and in

communication devices that support encryption. There is lingering suspicion, however, that the DES algorithm might contain a weakness that could allow the National Security Agency (NSA)—which has a vested interest in maintaining its ability to decrypt communications and which cooperated in the development of DES—to more easily break messages encrypted with DES.

RSA (which was named after its inventors) is an algorithm that was not developed by a government agency. Its creators—Rivest, Shamir, and Adleman—exploited the computationally difficult problem of factoring prime numbers to develop a nonsymmetric, or public key algorithm, which can be used for both encryption and digital signatures. *RSA* has since become a very popular alternative to *DES*. *RSA* is used by a number of software companies that produce products that must negotiate secure connections over the insecure Internet (such as web browsers), including Microsoft, Compaq, Sun, Netscape, and IBM.

In the mid '90s, the NSA and the Executive branch of the U.S. Government developed a new encryption technology and attempted to convince the rest of the government as well as the private industry to use it. First called "Clipper" and then "Capstone," the encryption scheme uses hardware that embodies a classified algorithm (called the "Skipjack" algorithm). The algorithm uses a secret key in combination with an escrow key or keys. The escrow keys are meant to be given to a pair of government or other key escrow agencies. Authorized law enforcement officials can then obtain the keys to the hardware device and decrypt any messages encoded by that device. The ostensible purpose is to make further digitally encrypted communications devices wiretappable. The Clipper and Capstone efforts have been abandoned because of a complete lack of support from the private sector because they are viewed as a threat to privacy.

The ciphers described here are not the only ones available for use in computers and networks today—other governments (such as the former USSR) were just as active as the United States in developing codes and ciphers, and many private individuals (especially in the last decade) have made contributions to the field of cryptography. *GOST* was developed in the former USSR, *FEAL* was developed by NTT Japan, *LOKI* was developed in Australia, and *IDEA* was developed in Europe. Most of these ciphers use patented algorithms that must be licensed for commercial use, but there are some (such as *Blowfish* and *Twofish*, which are described later in this chapter in the section on symmetric functions) that are not. Each cipher has strengths and weaknesses, some of which will be explored later in this chapter.

A recent addition to the ranks of encryption algorithms is the winner of a National Institute of Standards and Technology contest for a new Advanced Encryption Standard cipher to replace *DES*. The *Rijndael cipher* won the contest and can be expected to start showing up in all sorts of security products. This cipher was developed and selected using an open process, unlike *DES* and *Capstone*, which may eliminate some people's concerns about backdoors and hidden weaknesses in government-supported algorithms. The truly paranoid will most likely still avoid it merely because it has been blessed by the government.

Note All of the ciphers described in this section have the same weakness: If you know the cipher being used to encode a message but not the key, there are a number of attacks you can use to attempt to decode the message, including the "brute force" method of trying all of the possible keys.

The purpose of ciphers, after all, is to hide information. Hiding information would not be a useful activity (especially for wartime governments that have other pressing areas to spend time and money on) if no one were interested in the information being hidden. The converse of hiding information is attempting to discover what is hidden, and advances in breaking codes (or deciphering codes without the keys) have progressed hand-in-hand with developments in creating

codes. The practice of attempting to break codes is called *cryptanalysis*, and the people who break codes are called *cryptanalysts*.

Cryptanalysts have developed other key discovery methods beyond brute-force cryptanalysis. *Differential* and *linear cryptanalysis* are two examples of the esoteric art of breaking codes; they are complex mathematical analyses that would take a book each to explain. Some ciphers are more vulnerable to these two methods of cryptanalysis than other ciphers. Some ciphers use a longer key than others (a longer key contains more bits) and therefore require more time or compute power to go through all of the possible keys. Some ciphers can accept a variable number of bits in the key (you can choose how strong you want the encryption to be). Once the key is discovered, however, all of the messages encrypted using that cipher with that key are compromised.

The Almost Perfect Cipher

There is one encryption cipher—the *one-time pad*—which cannot be compromised without possession of the key, even with all the time left in the universe and all the compute power that is theoretically possible. It is not simply improbable that the key would be discovered or the message retrieved by using brute force; it is impossible. Unfortunately, the requirements of the cipher make it impractical for use in anything but certain kinds of low-bandwidth communications.

A one-time pad uses a key that is exactly the same length as the message being encoded. The key must be completely random (anything less than random leaves your message open to certain kinds of cryptographic analysis), and no portion of it can be re-used without compromising the security of your message. Each letter (or byte) of your message is combined mathematically with an equal-sized portion of the key (often by the XOR mathematical function or addition with modulus mathematical function), which results in the ciphertext and uses up the key.

The reason that the one-time pad is so secure is that, from the ciphertext being decoded, any resulting plaintext (of the same length) and associated key is equally likely. For example, "henryjtillman" encoded with the one-time pad key "lfpwxzgwpioeq" results in the ciphertext "tkkhsjafbavfe." While the ciphertext decoded with the correct key produces the original message, the ciphertext can also be decoded using the possible key "swgpnmquypciq" resulting in the message "andrewjackson," or using the key "gbywrwcmllkwz" resulting in the message "milkandcookie." The attacker has no way of knowing which key and resulting plaintext is correct.

The problem with the one-time pad is that it requires a key as big as the message being sent, and both the sender and the receiver must have the same key. If you must encrypt a 10Mbps Ethernet link you could use up a CD-ROM worth of key data in just 10 minutes!

Note Clearly, the one-time pad is best used in cases where communication is infrequent or uses very little bandwidth, such as e-mail messages that must have the most secure encryption possible.

Encryption in Your Network

The previous section took a look at how cryptography developed and how it works in a cursory manner. The next section, "A Conspiracy of Cryptographers," will examine the world of cryptography in greater detail, explore the advantages and disadvantages of specific algorithms, as well as give you a glimpse of how they can be broken. This section focuses on how you can use encryption in your network.

After all, what good is a secret decoder wheel if you have no secrets to keep? You can use encryption to protect the following types of network data:

- Private communications
- Secure file storage
- User or computer authentication
- Secure password exchange

You should encrypt any communications containing sensitive or proprietary information that go over an insecure medium such as radio, a telephone network, or the Internet. Use file system encryption to protect sensitive data when operating system features are not effective (when the hard drive has been removed or the operating system has been replaced).

Private Communications

The most common use for encryption with computers is to protect communications between computer users and between communications devices. This use of encryption is an extension of the role codes and ciphers have played throughout history. The only difference is that instead of a human being laboriously converting messages to and from an encoded form, the computer does all the hard work.

E-Mail

Many e-mail packages include the facility to encrypt an e-mail message before sending it; even those programs that do not can include encrypted text that comes from a separate encryption program such as Pretty Good Privacy (PGP). When you receive an encrypted e-mail message you can have the e-mail package decrypt it for you (if your e-mail supports encryption—basic LAN e-mail products such as Microsoft Exchange and cc:Mail often don't, but advanced packages such as Microsoft Exchange Server do) or you can use an external program to decrypt the message.

In order to encrypt or decrypt a message you must have the key to the message. You'll usually use the same key for messages to the same person, so if you regularly exchange encrypted mail with a lot of people, you'll have a lot of keys to keep track of. Your e-mail package or your encryption package can make keeping track of keys easier by storing your keys in key rings (files on your disk drive that keep track of your keys for you). The key rings are, of course, encrypted and protected by a key as well. The benefit of this is that you only have to remember one key.

Communications Links

Encryption can protect the communication between network devices as well as between computer users. Any time two network devices are communicating over an *insecure* medium (that is a medium that an intruder can listen in on), you should consider encrypting the communication. For example, you might encrypt the data transferred over a microwave link between buildings if sensitive data flows over the link.

Encryption can be expensive, however, either in terms of the processing power required to encrypt or decrypt data, or in terms of the cost of specialized hardware to do the same thing. The more data there is to encrypt (the greater the bandwidth of the encrypted link), the more processing power is required. So, while any modern personal computer can encrypt a 56Kbps modem link without slowing down the user's applications all that much, encrypting a 100Mbps fast Ethernet link will tax even the most powerful RISC and Pentium processors.

Dedicated encryption hardware also costs much more than regular unencrypted communications devices like Ethernet adapters and modems. For these reasons you probably would only encrypt those portions of your network that are exposed in a way that they are accessible to an eavesdropper.

Secure File Storage

Encryption isn't just for communication. It can also be used to protect data in storage, such as data on a hard drive. All Unix implementations, as well as Windows NT, have many sophisticated security features. You may have configured your OS to allow only authorized users to access files while the operating system is running, but when you turn your computer off, all those security features go away and your data is left defenseless. An intruder could load another operating system on the computer, or even remove the hard drive and place it in another computer that does not respect the security settings of the original computer.

Encryption Utilities

You can use encryption software to encrypt specific files that you want to protect, and then decrypt them when you need to access them. The encryption and decryption process can be cumbersome, however, and you may end up having to remember a lot of encryption keys. Using encryption in this manner can also easily cause you to leave behind files that contain sensitive information—temporary files or files that are erased but still present on the hard drive after you delete the original encrypted document. This is obviously not what you want.

Encrypted File Systems

A better approach to security is to have the operating system encrypt and decrypt the files for you. You can get encrypted file systems for Windows NT, and Windows 2000 ships with a disk encryption service. Most versions of Unix will encrypt all the files on your hard drive, even temporary ones created by the applications you use.

You must supply the cryptographic key when you start your computer, but otherwise you can treat the files on your hard drive as regular, unencrypted files. This doesn't protect your files from being accessed while the operating system is running—that is what the operating system security features are for—but it does keep the data safe, even if someone steals the hard drive.

User or Computer Authentication

In addition to keeping secrets (either stored or transmitted), encryption can be used for almost the opposite purpose—to verify identities. It's used to authenticate users logging on to computers, it's used to ensure that software you download from the Internet comes from a reputable source, and it's used to ensure that the person who sends a message is really who they say they are.

Logon Authentication

You may not know it, but your operating system already uses encryption. When you log on to a Microsoft operating system such as Windows 95 or Windows NT, the operating system does not compare your password to a stored password. Instead, it encrypts your password using a one-way cryptographic function, and then compares the result to a stored result. Other operating systems such as Unix and OS/2 work the same way.

This seems a roundabout way of verifying your identity when you log on, but there is a very good reason the operating system does it this way. By only storing the encrypted version of your password, the operating system makes it more difficult for a hacker who gains access to get all of the passwords in your system. One of the first things a hacker goes for in a compromised system (that is, one where the hacker has gotten at least one password) is that computer's password list, so that the hacker can get account names and passwords that may be valid on other computers in your network.

With a one-way cryptographic function, it's easy to generate an encrypted value from the password, but it's difficult or impossible to generate the password from the encrypted value. Since only the encrypted values are stored, even a hacker who has complete access to the computer can't just read the passwords. The best the hacker can do is to supply passwords one by one and see if they match any of the hashes in the password list. The hacker can run a program to do this instead of typing them all in by hand, but it can take a while if the users of the computer have chosen good passwords.

Digital Signatures and Certificates

One problem with Internet e-mail is that it was not designed with security in mind. Messages are not protected from snooping by intermediate Internet hosts, and you have no guarantee that a message actually came from the person identified in the e-mail's From: field. Internet newsgroup messages have the same problem: you cannot really tell whom the message actually came from. You can encrypt the body of the message to take care of the first problem, and digital signatures take care of the second.

Digital signatures are useful because, while anyone can check the signature, only the individual with the private key can create the signature. The difference between a digital signature and a certificate is that you can check the authenticity of a certificate with a certificate authority.

In Chapter 2, you read about the danger in downloading files from the Internet. Hackers can (and do) upload programs and utilities that contain Trojan horses, which can leave your network wide open to intrusion. One way to make sure that those programs and utilities are safe to download is to only download files that have certificates signed by a reputable certificate authority.

Note The ExploreZip worm virus outbreak is an example of a Trojan horse that was passed as an executable attachment to e-mail that claimed it was sent from associates of the recipient. A digital signature standard would have prevented its spread because users would have been warned that the file was not signed by the sender.

A software company can transmit a certificate along with a file you download that certifies the file as free of viruses and Trojan horses. You can check the certificate with a certificate authority you trust (VeriSign is the certificate authority that Microsoft uses; Thawte provides cryptographic certificates free for personal use like e-mail encryption) and hackers cannot forge the certificates. Certificates and digital signatures are explained in more detail later in this chapter.

Secure Password Exchange

When you log on to your network file server, or when you connect to your Internet service provider, you supply a username and password. These two pieces of information control your access to the network and represent your identity on the network. They must be protected from eavesdropping.

Most network operating systems (Windows and all modern versions of Unix included) protect your

username and password when you log on by encrypting the username and password before sending them over the network to be authenticated. The file server (or ISP host) checks the encrypted username and password against the list of legitimate users and passwords. The host can check the password either by decrypting it and checking the database of passwords stored in the clear, or it can encrypt the stored password and check the result against what has been sent from the client over the network.

To keep the same encrypted data from being sent every time, the client can also include some additional information such as the time the logon request was sent. This way your network credentials are never sent unprotected over your local LAN or over the telephone system. Windows does accept unencrypted passwords from older LAN Manager network clients, however, so you should be careful about allowing older clients on your network.

Note Not every authentication protocol encrypts the username and password. SLIP, for example, does not. Telnet and FTP do not. PPP may, if both the dial-up client and server are configured that way. Windows by default requires encrypted authentication.

A Conspiracy of Cryptographers

Cryptography is the study of codes and ciphers. Like any other group of specialists, cryptographers have a language of their own to describe what they do. You don't have to be a theoretical mathematician to evaluate and use cryptography in your network, but it helps to have a general idea about ciphers when you are evaluating cryptography options for your network. Before we continue, there are a few terms you should know the precise way in which these terms are used in this book:

- **Algorithm** Detailed steps for performing a function.
- **Asymmetric Algorithm** An algorithm in which different keys are used for encryption and decryption. Public key and password hash algorithms are asymmetric.
- **Block Cipher** A cipher designed to operate on fixed-size blocks of data.
- **Breakable** A cipher that, given a reasonable amount of time and resources, can be compromised by a competent cryptanalyst.
- **Computationally Secure** A cipher that, given all the computational power that will be available to the most powerful governments for the foreseeable future, is unlikely to be compromised.
- **Keyspace** The range of all possible keys for a cipher. A cipher with a large keyspace is harder to crack than one with a smaller keyspace because there are more keys (numbers or combinations of letters) to try.
- **Restricted Algorithm** An algorithm that is kept secret to make it more difficult to break.
- **Secure** A cipher that, even given a reasonable amount of time and resources, most likely cannot be compromised by a competent cryptanalyst.
- **Stream Cipher** A cipher designed to operate on a continuous stream of data.
- **Strong** A cipher that, given the computational power that may reasonably be brought to bear on it any time in the near future, is unlikely to be compromised.
- **Symmetric Algorithm** An algorithm in which the same key is used for encryption and decryption. Private key algorithms are symmetric.
- **Unconditionally Secure** A cipher that, given an unlimited amount of time and an infinitely powerful processor, cannot be compromised.

Algorithms

When you encode a message (transform a plaintext into a ciphertext), you must perform a series of steps. For example, to encode a message using the PigLatin99 cipher, you do the following:

1. Take the first word of the plaintext.
2. Move the first consonant of the word (if the word starts with a consonant) to the end of the word and follow it with "ay."
3. Write the modified text down (this is the ciphertext).
4. If you have words left, take the next word of the plaintext and then go to step 2. Otherwise, you're done.

A set of instructions, like the steps outlined above, is an *algorithm*. Ciphers are algorithms that describe how to encode or decode messages; therefore, ciphers are *cryptographic algorithms*. Not every algorithm is a cipher, of course—algorithms are used for many other things, especially in computers. In fact everything that computers do is detailed in an algorithm of one sort or another.

Algorithms can be performed by people (recipes are algorithms), but the all-time-champion algorithm followers are computers. This, combined with the fact that most ciphers are meant to be performed by a computer, means that most ciphers are detailed in computer languages such as C rather than in English steps like the example above. The following is an example of a cipher in C:

```
/*
ROT(n)
Usage: rot [e|d] number inputfile outputfile
The number should be between 0 and 26, exclusive
Assumes a contiguous linear character encoding (i.e. ASCII)
*/

#include <stdio.h>
#include <string.h>

int do_rotate( char isencrypt, int key, int thechar ){
    if(isencrypt=='d')
        key *= -1;
    if( thechar >= 'A' && thechar <= 'Z'){
        thechar+=key;
        if(thechar>'Z')
            thechar-=26;
    }
    if( thechar >= 'a' && thechar <= 'z'){
        thechar+=key;
        if(thechar>'z')
            thechar-=26;
    }
    return(thechar);
}

void main(int argc, char *argv[]){
    FILE *infile,*outfile;
    int thischar, keyval, isencrypt;

    isencrypt=(char)argv[1];
    keyval = atoi(argv[2]);
    if(keyval>0&&keyval<26&&(isencrypt=='e' || isencrypt=='d')){
        if((infile = fopen(argv[3], "rb")) != NULL){
            if((outfile = fopen(argv[4], "wb")) != NULL){
                while((thischar = getc(infile)) != EOF){
```

```

        putc(do_rotate(isencrypt, keyval, thischar), outfile);
    }
}
}
}
}

```

If you examine the preceding cipher, you will notice that there are two basic parts to it: the main part that takes care of opening, reading, stepping through, and writing files, and the `do_rotate` part that performs a specific function (in this case, adding or subtracting the key value) to a portion of the file (in this case, one character). This is the function that is performed by the cipher. (It is also defined as a function written in the C programming language, but that's beside the point. If it were written in BASIC it might be called a sub-routine, but it still performs a cryptographic function.)

You can implement other ciphers by replacing the `do_rotate` function with a different function, such as one that performs an XOR on the character with a specific key. You would then be implementing an XOR cipher instead of a ROT cipher. There are many different ciphers that you can implement in this manner, each performing a specific mathematical function on input text (plaintext) with a key to produce the output text (the ciphertext).

Note XOR is the mnemonic for a logical "exclusive-or," one of the simplest mathematical operations that a computer can perform. An XOR is an operation that compares two bits and returns a 1 if they are the same or a 0 if they are different. Using this operation, you can easily "transform" one bit stream with another one in a manner that is reversible by performing the same operation again with the same key. It's an obvious candidate for simple encryption for that reason.

Symmetric Functions

If the same key can be used to encrypt or decrypt the message (as in the Rot(13) example in the Algorithms section), then the cipher uses a symmetric function. Both the sender and receiver must have that same key. Good symmetric ciphers are fast, secure, and easy to implement using modern microprocessors.

Some ciphers are more secure than others. The XOR cipher, for example, is not very secure. A competent cryptanalyst can decode an XOR-encoded message in short order. There are two general features of a symmetric algorithm that make it secure:

- The algorithm produces ciphertext that is difficult to analyze.
- The algorithm has a sufficiently large keyspace.

Cryptanalysts test ciphertext for correspondences in the text, an uneven distribution of instances of numbers, and essentially anything that differentiates the ciphertext from a series of truly random numbers. A good algorithm will produce a ciphertext that is as close to random in character as possible. This is where the XOR cipher fails miserably; an XOR-ed message has a lot in common with a regular ASCII text message. Cryptographers will exploit these commonalities to recover the key and decode the whole message.

A cryptanalyst who cannot exploit the nonrandom in ciphertext has little choice but to simply try all the possible key combinations to decode the message. This is a lot like the hacker trying to guess the password to your system—if they don't know that the password is a birthday or the name of your dog, then they must try all the possible passwords.

Note Just as a longer password is safer than a shorter one, a longer key is more secure than a shorter key.

There are a number of symmetric ciphers used in both software and hardware. You can get a feel for what is available by comparing the following ciphers.

Data Encryption Standard (DES)

IBM and the U.S. National Security Agency cooperated to develop this cipher (see the "How to Keep a Secret" section earlier in this chapter). It has been designed to be resistant to differential cryptanalysis, but has been shown to be susceptible to linear cryptanalysis (linear and differential cryptanalysis are described later in this chapter in the section on "Attacks on Ciphers and Cryptosystems."). Its key length is only 56 bits, which makes it increasingly susceptible to a brute-force examination of all of the possible keys for an encrypted ciphertext. DES is an ANSI standard commonly found in encryption hardware and software.

International Data Encryption Algorithm (IDEA)

This cipher has a key length of 128 bits—considerably more than DES uses. While a sufficiently motivated and financed organization can break a DES-encoded message, the large key space makes a brute force attack on IDEA impractical. IDEA was designed to be immune to linear and differential cryptanalysis, and you can reasonably be assured that not even the NSA can decode an IDEA-encrypted message without the key. IDEA is patented both in Europe and United States.

Rijndael

Rijndael (the new United States standard for encryption) has three settings for key length. It can be configured to use 128, 192, or 256 bits for the key and (independently) for the block size. Rijndael is likely to become widely implemented, so products that support it are more likely to interoperate with other products.

Blowfish/Twofish

These ciphers can use a key with anywhere from 32 to 448 bits (for Blowfish) or 128, 192, or 256 bits (for Twofish), allowing you to select how secure you want to make your message. They were designed to be immune to linear and differential cryptanalysis. Their developer, Bruce Schneier, has not sought a patent on the algorithms so that a good, freely implementable algorithm would be available to both private individuals and the public sector.

Asymmetric Functions

When you type your password to log on to Windows, it is encrypted and compared against the stored encrypted value of your password (see the section "Logon Authentication" earlier in this chapter). The reason the password is stored using an asymmetric function (also called a one-way function, hash, trap-door, digest, or fingerprint) is to make it difficult for a hacker or other network intruder to determine your password, even if the hacker has gained access to the operating system's stored settings.

Hash functions can also be used for other purposes. For example, you can use a hash to "fingerprint" files (create a digital fingerprint or hash that is unique to that file). A hash function can produce a result that is much smaller than the input text; a hash of a multi-megabyte word-processor document, for example, may result in a 128-bit number. A hash (or fingerprint) is

also unique to the file that produced it; it is practically impossible to create another file that will produce the same hash value. You might use this kind of hash to make sure that your Internet-distributed software product is delivered free of viruses and other malicious modifications. You can allow your customers to download the software, and then tell them what the hash value for the software files is. Only your unmodified software files will hash to the same value.

Note One feature of a hash function (especially one that produces short hashes) is that any hash value is equally likely. Therefore, it is practically impossible to create another file that will hash to the same value. Many open-source projects use hash functions that you can download along with software to ensure that the software hasn't been modified and then reposted by a hacker.

Some hash functions require a key, others do not. Anyone can calculate a hash that does not use a key; this kind of hash is good for distributing software or making sure that files have not been changed without you noticing. A hash function with a key can only be calculated by someone (or something) that has the key.

Public Key Encryption

While symmetric ciphers use the same key to encrypt and decrypt messages (that's why they're called symmetric), public key encryption (or a public key cipher) uses a different key to decrypt than was used to encrypt. This is a relatively new development in cryptography, one that solves many longstanding problems with cryptographic systems, such as how to exchange those secret keys in the first place.

The problem with symmetric ciphers is this: Both the sender and the recipient must have the same key in order to exchange encrypted messages over an insecure medium. If two parties decide to exchange private messages, or if two computers' network devices or programs must establish a secure channel, the two parties must decide on a common key. Either party may simply decide on a key, but that party will have no way to send it to the other without the risk of it being intercepted on its way. It's a chicken-and-egg problem: Without a secure channel, there is no way to establish a secure channel.

In 1976, Whitfield Diffie and Martin Hellman figured a way out of the secure channel dilemma. They found that by using a different key, the one-way function could be undone. Their solution (called public key cryptography) takes advantage of a characteristic of prime and almost prime numbers—specifically, how hard it is to find the two factors of a large number that has only two factors, both of which are prime. Since Diffie and Hellman developed their system, some other public key ciphers have been introduced. For example, the difficulty of determining quadratic residues (a subtle mathematical construct that few people other than mathematicians and cryptologists really understand) has been exploited to make a public key cipher.

With a public key cipher, one key (the public key) is used to encrypt a message, while the other one (the private key) is the only key that can decrypt the message. This means that you can tell everyone your public key, even complete strangers and NSA agents. Anyone who has your key can encrypt a message that only you can decrypt. Even the NSA agent who has your public key cannot decrypt the message.

One problem that plagues secure public key ciphers is that they are slow—much slower than symmetric ciphers. You can expect a good public key cipher to take 1,000 times as long to encrypt the same amount of data as a good symmetric cipher. This can be quite a drag on your computer's performance if you have a lot of data to transmit or receive.

Although it is much slower than symmetric systems, the public key/private key system neatly solves the problem that bedevils symmetric cryptosystems. When two people (or devices) need to establish a secure channel for communication, one of them can just pick a secret key and then encrypt that secret key using the other's public key. The encrypted key is then sent to the other party, and even if the key is intercepted, only the other party can decrypt the secret key using their private key. Communication may then continue between the two parties using a symmetric cipher and that secret key. A system that uses both symmetric and public key encryption is called a *hybrid cryptosystem*.

Protocols

There is a lot of mathematics in cryptography, and to be a good cryptographer you almost have to be a theoretical mathematician, but there is more to cryptography than mathematics. There is even more to cryptography than ciphers that use math. The ciphers all by themselves are merely a computational curiosity—they only become useful when secure communications systems are built using them.

Communications systems are built out of protocols. Protocols, in turn, describe how the entities in a communications system (such as the computers in a network) cooperate. Protocols also perform the same function in human terms. Robert's Rules of Order, for example, contains the protocols for people communicating in a meeting or assembly, including who may speak at any particular time and how to excuse yourself from the meeting.

For a computer, the protocols describe who will transmit when, what will be transmitted, how it will be formatted, and what (and who) will transmit next. A protocol is very much like an algorithm, but while an algorithm describes the steps for one computer to take, a protocol describes the steps that two or more communicating computers must take.

Note The parts of a protocol that one computer performs comprise an algorithm. You can think of a protocol as a set of algorithms that communicating computers use.

Every time you log on to your network or connect to the Internet, you use protocols. The Internet uses the TCP/IP protocol suite, which is a set of protocols that all communicating Internet computers use. Windows NT uses the NetBIOS protocol along with either the TCP/IP, NWLink, or NetBEUI protocols to provide access to shared files and printers. These are general-purpose protocols that allow networked computers to communicate, and they are designed more for efficiency and scalability than for security.

Cryptographers have a very narrow view of protocols. A cryptographer will consider such aspects as the computation time required for a protocol, or whether or not it is feasible for a hundred thousand computers to use it, but the cryptographer is more concerned with whether the protocol is susceptible to such network malfeasance as the man-in-the-middle attack (see the "Attacks on Ciphers and Cryptosystems" section).

The following is a simple protocol for two people (let's call them Alice and Bob, in the tradition that real cryptographers follow) to establish a symmetrically encrypted communications channel using public key encryption. Here are the steps:

1. Alice sends Bob her public key (or Bob gets Alice's public key from a public key directory).
2. Bob selects a secret key for use with a symmetric cipher, encrypts it using Alice's public key, and then sends it to Alice.
3. Alice decrypts the secret key using her private key.

4. Alice and Bob use the secret key and the symmetric cipher to send private messages to each other.

However, this protocol has a problem. We'll explore this problem in the next section.

Attacks on Ciphers and Cryptosystems

Throughout history, cryptography has been a race between those trying to keep secrets and those trying to find out what those secrets are. These code makers and the code breakers are often the same people. It's no different today. If you are really serious about keeping information private using cryptography, you should be aware of what others may do to ferret out that private information. Cryptographers call these attempts *attacks*, and they come in two forms: cryptanalysis and protocol subversion.

Cryptanalysis Techniques

As we noted earlier, cryptanalysis is the process of examining encrypted information to try to determine what the encrypted message is, or what the key that encrypted the message is (which, of course, gives you what the message itself is). There are a number of different techniques a cryptanalyst can use, including those discussed here:

Keyspace Search This is the hard way to crack an encrypted message. A keyspace search involves checking all of the possible keys that might have been used to encrypt the message. This is like trying all of the possible combinations on a bank vault in order to open it. A keyspace search is only feasible when there are not very many possible keys. A cryptanalyst might use this technique if the key length is 32 or 48 bits, and perhaps if it were 56 bits and the message were really worth the million-dollar hardware that would be required. Keyspace searches of larger keyspaces are impractical at the present level of computing technology.

Known Plaintext A cryptanalyst can reduce the number of possible keys to be searched for many ciphers if the cryptanalyst already has the plaintext of the encrypted message. (Why would the cryptanalyst want the key if the message is already out? Perhaps there is another message encrypted with the same key.) If even a portion of the message is always the same, especially at the beginning of the message (for example, the headers in an e-mail message are always the same), your ciphertext may be vulnerable.

Linear and Differential Cryptanalysis A cryptanalyst may also look for mathematical patterns in collections of ciphertexts that have all been encrypted with the same key. Some ciphers (not all) are vulnerable to either or both of these kinds of analysis, and a cryptanalyst may then have a much smaller range of keys to search.

Protocol Subversion

Just choosing a good cipher doesn't make your network safe. Recall the protocol example in the previous section with Alice and Bob. They both used a pretty good public key encryption system to exchange a secret key that they then used with a very secure symmetric cipher. That wasn't enough to ensure their privacy, though, because their protocol was weak. Consider the following scenario with a determined and powerful adversary, Mallet:

1. Alice sends Bob her public key.
2. Mallet intercepts Alice's public key and sends Bob his own public key (alternatively, Mallet intercepts Bob's request for Alice's public key from the directory and substitutes his own).

3. Bob selects a secret key for use with a symmetric cipher, encrypts it using Mallet's public key (which he thinks belongs to Alice), and then sends it to Alice.
4. Mallet intercepts Bob's response. He then encrypts the secret key he received from Bob with Alice's own public key and sends it to her.
5. Alice decrypts Mallet's secret key using her private key.
6. Mallet maintains the illusion that Alice and Bob are talking to each other rather than to him by intercepting the messages and retransmitting them.

Obviously, this man-in-the-middle attack scenario requires that Mallet be able to intercept messages and substitute his own. This is possible in some media (such as the Internet) and not others (such as radio). A protocol can be modified to be resistant to man-in-the-middle attacks. Consider the following modification (called the Interlock Protocol and invented by Ron Rivest and Adi Shamir) and how it affects Alice and Bob's negotiation of a secure key:

1. Alice sends Bob her public key.
2. Bob sends Alice his public key.
3. Alice selects a secret key and sends Bob a message encrypted with that secret key. Neither Bob nor Mallet can read the message, since neither of them have the secret key.
4. Bob selects a secret key and sends Alice a message encrypted with that secret key. Neither Alice nor Mallet can read the message, since neither of them have the secret key.
5. Alice encrypts her secret key with Bob's public key and then sends it to Bob. Bob can then read Alice's message.
6. Bob encrypts his secret key with Alice's public key and then sends it to Alice. Alice can then read Bob's message.
7. Bob and Alice continue to communicate, using the secret keys they have established.

The above protocol is not perfect, but it does make it much more difficult for Mallet to impersonate Alice and Bob. Mallet cannot read Alice's message until step 5, and cannot read Bob's message until step 6. In both cases, it is too late for Mallet to change the message already sent to the other party. In order for Mallet to subvert this protocol, he must correctly guess what is being sent in steps 3 and 4 in order to create a convincing counterfeit.

Digital Signatures

In the above examples using public key encryption, the message is encrypted with the public key and decrypted with the private key. You can also do it the other way—encrypt with the private key and decrypt with the public key.

Why would you want to encrypt a message that anyone can decrypt? That seems a bit silly but there is a good reason to do so; only the holder of the private key can encrypt a message that can be decrypted with the public key. This encryption is in effect a digital signature, proving that the holder of the private key produced the message.

Since the purpose of a digital signature is not to conceal information but rather to certify it, the private key is often used to encrypt a hash of the original document, and the encrypted hash is appended to the document or sent along with it. This process takes much less processing time to generate or verify than does encrypting the entire document, and it still guarantees that the holder of the private key signed the document.

Steganography

Cryptography can be very effective at keeping a secret. With a sufficiently powerful cipher and a sufficiently long key, even major world governments cannot read your diary. What if you don't want people to know that you're keeping secrets? After all, an encrypted file or an encrypted hard drive is pretty strong evidence that you're hiding something. *Steganography* is the process of hiding those encrypted files where it is unlikely that anyone will find them.

Encrypted files look like random numbers, so anything that also looks like random numbers can be used to hide an encrypted message. In graphics images that use many colors, the low-order bit for each pixel in the image doesn't make much difference to the quality of the image. You can hide an encrypted message in the graphic file by replacing the low-order bits with the bits from your message. The low-order bits of high-fidelity sound files are another good place for encrypted data. You can even exchange encrypted messages with someone surreptitiously by sending graphics and sound files with those messages hidden in them.

Random Sequence Generation

Most cryptographic systems manage the selection of keys and the negotiation of protocols for you. Systems that do this must be able to select keys that are not easily guessed, because one way to attack a cryptographic system is to predict the keys that might be used in the system. These keys are selected by generating random numbers.

It is difficult for a computer to generate good random numbers. Computers, by their very nature, are extremely predictable, and hundreds of thousands of engineers have labored (collectively) millions of years to make them more so. If you run a computer program twice and give it the same input the second time as you did the first, you will get the same output the second time as you did the first. Since the whole point of a truly random number is not to be able to guess the output based on the input, computers (unassisted) make lousy dice-throwers.

The best that computers can do by themselves is good *pseudorandom numbers*. Pseudorandom numbers are created by a deterministic means (that is, given identical starting conditions, identical numbers will be produced). Good pseudorandom numbers have a long periodicity and satisfy the other conditions of random numbers, such as incompressibility and having an even distribution. Random numbers, on the other hand, are unpredictable (a series of random numbers cannot be reproduced, even from identical starting conditions).

In order to get a good random number (to use as a seed value, for example) the computer must look outside itself because computers are inherently deterministic. There are many sources of randomness in the real (non-computer) world—the weather, ocean waves, lava-lamp wax gyrations, the times between one keystroke and the next—and a computer can measure these events and use them to generate random numbers. Keystroke timing is commonly used to generate secret keys. Another way is to ask the user to type in a paragraph or two of text; there are no published algorithms that will predict arbitrary user input (yet).

If a random number is going to be used as a seed for pseudorandom numbers, it should have enough bits to make it difficult to guess. For example, you don't want to protect a 128-bit cryptosystem that uses IDEA with a password of 8 characters or less for a seed—this is effectively only about 48 bits of security if you just use printable ASCII characters in the password.

Case Study: Encryption Messages

While working for a network services company as a network integrator, I received a request for help from a government contractor who had been hired to decrypt the contents of a hard disk that had been seized by a federal law enforcement agency. There are two kinds of encryption in common use on computers today: encryption that will keep coworkers out of your files, and encryption that will keep everyone out of your files.

An examination of the contents of the hard drive revealed that this drive (which contained an encrypted NTFS volume protected by a pass-phrase that we did not have) used the latter kind of encryption. I had to inform the contractor that our organization did not have the resources that were required to break the encryption, and it was likely that no organization would.

In a related incident, the same contractor asked us to take a look at another server with an encrypted disk. (This contractor had won a bid to provide cryptographic services to the government for a slew of related cases.) The disk encryption was remarkably similar in quality, and just as difficult to crack—but this server came with a backup tape left in the DAT drive that contained the entire contents of the server in unencrypted form.

Part II: Firewall Technology

Chapter List

Chapter 6: Packet Filtering

Chapter 7: Network Address Translation

Chapter 8: Application–Level Proxies

Chapter 9: Virtual Private Networks

Chapter 10: The Ideal Firewall

Chapter 11: Configuring a Real Firewall

Part Overview

Topics Covered:

- How packet filtering works
- Problems with packet filtering
- How Network Address Translation works
- Problems with Network Address Translation
- How proxies work
- Problems with proxies
- How internet tunnels and Virtual Private Networks work
- Problems with tunnels and VPNs
- How to determine the level of security your network requires
- How to implement your required level of security

Chapter 6: Packet Filtering

Overview

Packet filters were the original firewalls. The first attempts to make TCP/IP secure were based on the idea that it's pretty easy for a router to inspect the header of TCP/IP packets and simply drop packets that don't conform to the specifications you want to accept.

Packet filters have problems that make them insufficient to provide total security for an internal network. They are now combined with proxy servers and Network Address Translators to solve those problems.

Proxy servers were originally designed to make the World Wide Web faster. Network Address Translators were originally designed to increase the address space available to private organizations and to solve IP address numbering problems associated with attaching existing private TCP/IP networks to the Internet. The serendipitous security benefits of both of these functions were integrated with packet filtering and encryption technology to create the modern effective firewalls in use today.

Neither proxy servers nor Network Address Translators can be properly secured without a packet filter, and a packet filter cannot provide total security without the services of a proxy server or a Network Address Translator. Because these services must be combined into a single coherent security function to be effective, you should use firewalls that make effective use of all three methods to truly secure your network.

There are two primary types of packet filtering:

- Original, or "stateless" packet filtering, often used by routers and operating systems.
- Stateful inspection packet filtering, used in all modern firewalls.

This chapter discusses "pure" packet filtering: packet filtering not combined with proxy or Network Address Translation functions. Pure packet filters are still in use all over the place, so this discussion remains very current.

How Stateless Packet Filters Work

Packet filters are border routers that increase security by determining whether or not to forward a packet based on information contained in the header of every individual packet. Filters can theoretically be configured to determine this based on any part of the protocol header, but most filters can be configured only to filter on the most useful data fields:

- Protocol type
- IP address
- TCP/UDP port
- Fragment number
- Source routing information

The following sections detail each of these fields.

Protocol Filtering

Protocol filtering filters packets based on the content of the IP protocol field. The protocol field can be used to discriminate against entire suites of services, such as:

- UDP
- TCP
- ICMP
- IGMP

For example, if you have a single-purpose server serving a TCP-based service like HTTP, you could filter out all UDP services. Unfortunately, the protocol field is so general (only four common protocols are available to filter on) that most servers and routers will have to leave all of them open.

IP Address Filtering

IP address filtering allows you to limit connections to (or from) specific hosts and networks based on their IP address. Most filters allow you to either deny access to all hosts except an accepted list or allow access to all hosts except a denied list.

Specific denial of certain hosts is almost worthless, since you'd have to keep track of every hacker that had ever attacked your network and assume that they have no way to gain information from a different IP address, which they always will. Relying on specific denial is not a strong security policy.

Specific acceptance of certain host addresses provides particularly strong security, however; it is the strongest form of security that a stateless packet filter can provide. By denying access to all hosts except a list of known IP addresses, you can ensure that your routers can only be reached by the IP addresses of machines or networks you know about. This list could be other networks in your organization, the networks of your customers, or the networks of work-at-home users. By denying access to all other IP addresses, you make it nearly impossible for a hacker to exploit your network. To hack into your network, a hacker would have to have access to your list of allowed IP addresses.

It is possible for hackers to use source routing, (explained in detail in the next section), to "spoof" IP addresses. Source routing would allow a hacker to put an allowed address into a packet and then capture the return by specifying that responses be routed back to his computer. For this reason, packet filters should always be configured to drop sourcerouted packets.

Good packet filters will allow you to specify hosts on a per-protocol basis, so (for example) you could allow all hosts to access TCP port 80 for HTTP service but only hosts from your company network to access TCP port 23 (Telnet). Most simple filters don't have allowed lists per protocol, so you can only assign a single list of hosts allowed for all protocols.

It's important to remember that a filter can only limit addresses based on the IP address field's contents, which could be different than the actual source host. It's easy for hackers to forge the IP address field of a packet, so it's certainly possible for them to get a packet past a packet filter if they know an address that the filter will pass. This would be useful in instances where a round-trip is not necessary, such as in a denial-of-service attack or where the return address for the protocol is contained in the payload of the packet as well as the header (as in FTP).

TCP/UDP Ports

TCP or UDP port information is the most commonly used information to filter on because this data field indicates most specifically what the packet is for. Port filtering is also commonly referred to as protocol filtering because the TCP or UDP port number identifies higher-level protocols. Figure 6.1 shows how a stateless packet filter discriminates based on TCP or UDP port number.

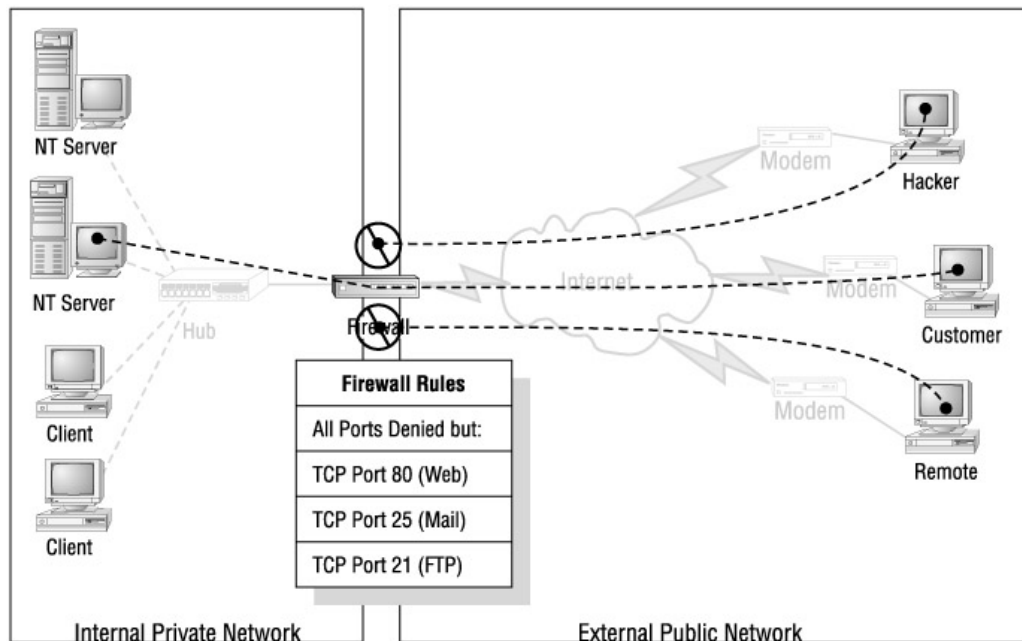


Figure 6.1: A packet filter rejects undesired traffic.

Common protocols that can be filtered based on the TCP or UDP port field are:

Daytime	DNS	NetBIOS Session
Echo	HTTP	IMAP
Quote	Gopher	NFS
FTP	POP	Whois
Telnet	SNMP	RSH
SMTP	NNTP	

As with IP addresses, most packet filters allow you to either pass all protocols except a denied list or pass no protocols except an allowed list, and as with IP addresses, passing no protocols except an allowed list is more secure. Unlike IP address filtering, blocking only certain ports is still useful because most hacking exploits target only a few specific protocols. The most important protocols to block are listed here:

- **Telnet** Leaving this port open on a host will allow hackers to open a command prompt with a large amount of access to your machine.
- **NetBIOS session** Leaving this port open to the Internet on Windows or SMB serving hosts will allow hackers to attach to your file servers as if they were local clients.
- **POP** You should implement a VPN connection for remote clients who need to check their mail because POP uses plaintext passwords to allow access, which will allow hackers to sniff user's passwords off the network.
- **NFS** Unix clients should not leave open access to NFS ports for the same reason Windows clients should not leave open access to the NetBIOS ports.

- **X Windows** Running X client software (the terms "client" and "server" in X environments have the opposite of their usual meanings) will leave your server vulnerable to attack.
- **Windows Terminal Services** Exposing Windows Terminal Services to the Internet means your terminal server is protected by just a username and password. There are numerous ways to extract that information from clients on your network.

These ports are especially sensitive to attack because of the high level of functional control they give the attacker. Other ports, like DNS, could be used to damage some specific information, but the service itself is not rich enough to control the machine directly and are therefore of less value to attackers (of course, all listening services are potentially vulnerable to buffer overrun and related exploits).

Other ports you should block include any sort of remote access or remote control software such as pcAnywhere or VNC.

Filtering on Other Information

In addition to the standard fields, headers contain other information that can be used to determine whether or not a packet should be passed.

Source routing and fragmentation are two techniques supported by the IP protocol that are largely obsolete and frequently exploited by hackers. Most packet filters will allow you to simply drop any packets that are source routed or fragmented.

Source Routing

Source routing is the process of defining the exact route a packet must take between hosts in an IP connection. Source routing was originally used for debugging and testing purposes, but it is now frequently used by hackers who can put any address in the source field and still ensure the packet will return by specifying her own machine in the source route.

Two types of source routing exist:

- Loose source routing, which indicates one or more hosts the packet must flow through, but not a complete list.
- Strict source routing, which indicates the exact route a packet must follow between hosts.

Of the two types, loose source routing is most often used by hackers because the he can simply plug in the IP address of his machine to make sure the packet comes back to him by any means.

Unless you use source routing in your network, configure your filters to drop any source-routed packets. No protocol or ISP requires source routing.

Fragmentation

Fragmentation was developed to support the passage of large IP packets through routers that could not forward them due to the frame size constraints encountered in some early networks. Fragmentation gave any router in the path between two hosts the ability to chop up an inbound IP packet into multiple smaller packets and then forward them on size-constrained networks. The receiving system simply waited for all fragments of the packet to reassemble it to its original form.

The problem with fragmentation comes from the fact that the most useful filter data, the TCP or

UDP port number, is only provided in the beginning of an IP packet, so it will only be contained in fragment 0. Fragments 1 and higher cannot be filtered based on port information because they don't contain any port information. So most early filters simply forward all subsequent fragments with the assumption that if the 0th fragment had been dropped, the subsequent fragments would be worthless.

But that's not always the case. Many flawed versions of TCP/IP running on internal hosts might reassemble the packet anyway, and if the 1 through n packets contained a valid TCP packet, they'd go ahead and use it. This meant that a hacker could modify his IP stack to start all fragment numbers at 1 and effectively bypass the filter altogether.

Problems with Stateless Packet Filters

Packet filters suffer from two problems that prevent them from being completely effective:

- They cannot check the payload of packets.
- They do not retain the state of connections.

These problems make packet filters alone insufficient to secure your network.

No Service-Specific Security

Packet filters make pass/drop decisions based solely on header information; they do not inspect content for the presence of dangerous or malformed data to determine whether or not that data should be passed. For this reason, packet filters alone do not constitute effective security.

For example, HTTP content flowing back into your network could contain Trojan horses embedded in ActiveX controls. Your packet filter cannot determine this, so it simply passes the content through. Or you may allow SMTP port 25 through to your mail server to receive e-mail, but the filter can't determine that a malformed e-mail passing through it will crash your e-mail server.

Service-specific security can only be implemented by the service-specific filters used by proxy servers and true firewalls.

No Connection State Security

Most packet filters are stateless—that is, they do not retain information about connections in use—they simply make pass/drop determinations packet by packet and based only on the information contained within that packet. Stateless packet filters cannot determine whether or not to drop fragments because they retain no information about the fragment's service port. Stateless packet filters also cannot determine when a return socket connection applies to a connection established from inside the network, so they must be configured to simply pass all TCP ports in the range of a normal return socket. For this reason, many early packet filters simply pass all TCP ports above 1024.

Modern port filters and all modern firewalls use state information to keep track of connection status and thereby more positively control the routing of packets through your network.

If you can't use proxy servers to eliminate routing at your border, use state-based packet filtering and Network Address Translation.

OS Packet Filtering

Most modern operating systems, Unix and Windows included, include packet filtering as part of the TCP/IP stack. This means that you can configure unique packet filtering rules for each server based on its individual function. This is called end-system packet filtering because the final computer in the route, the host to which the packet is actually addressed, performs the filtration.

Intermediate systems like packet filtering routers and firewalls can be configured to drop or pass packets based on the ultimate address, so end-system packet filtering may seem unnecessary. But no border system can protect your server from an internal attack or from an attack that somehow sidesteps your border security by exploiting an improperly secured VPN or dial-up connection.

By including backup packet filtering directly on servers, you can provide an extra level of security that will still be in place if your border security fails or if the attack comes from inside your network.

You should use the packet filtering functionality of your server's operating systems to guarantee that you are only serving those protocols you intend to serve publicly. Hostbased packet filtering allows you to ensure that each server exposes only those services you intend.

How Stateful Inspection Packet Filters Work

Standard packet filters have a number of flaws, all of which stem from the fact that a single packet in a communication does not contain enough information to determine whether or not it should be dropped, because it is part of a larger communication. Stateful inspection packet filters solve this problem by retaining the state of all the communication flowing through the firewall in memory, and using that remembered state to determine whether or not individual packets should be dropped. Stateful inspectors filter entire communication streams, not just packets.

Stateful packet filters remember the state of connections at the network and session layers by recording the session establishment information that passes through the filter gateway. The filters then use that information to discriminate valid return packets from invalid connection attempts or hacking.

Most stateless packet filters simply allow all ports above 1024 to pass through the firewall because those ports are used for the return sockets of connections initiated inside the firewall. This is extremely poor security—nothing prevents Trojan horses from waiting inside your network on a service port above 1024, so stateless packet filters cannot prevent this sort of intrusion.

Stateful packet filters, on the other hand, do not allow any services through the firewall except those services they're programmed to allow and connections that they already maintain in their state tables.

When a trusted internal host connects to a TCP socket on an external untrusted host, it transmits with the connection synchronization packet the socket (IP address and port) on which it expects to receive a response. When that SYN packet is routed through the stateful inspection filter, the filter makes an entry in its state table containing the destination socket and the response socket, and then forwards the packet onto the untrusted network. When the response comes back, the filter can simply look up the packet's source and destination sockets in its state table, see that they match an expected response, and pass the packet. If no table entry exists, the packet is dropped because it

was not requested from inside the network. Figure 6.2 shows the establishment phase of a stateful filter.

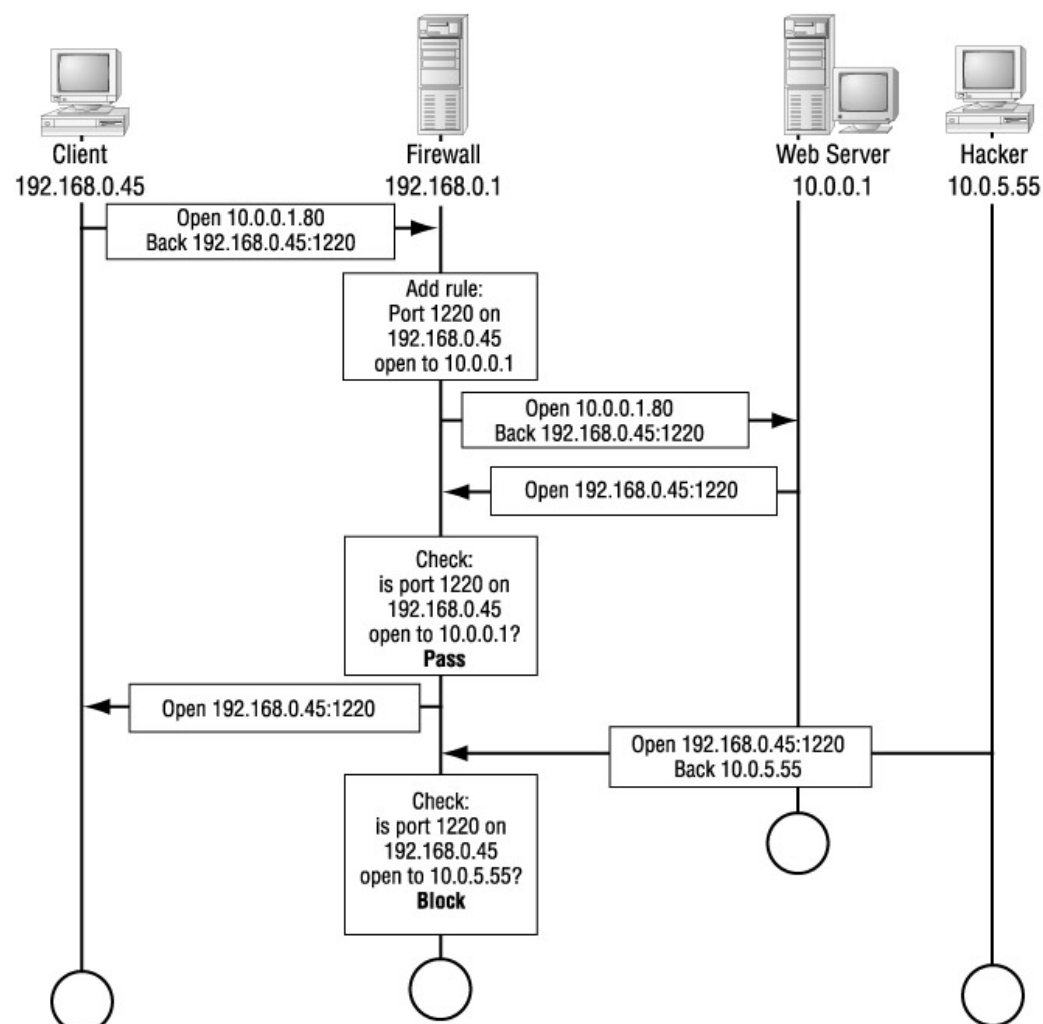


Figure 6.2: A stateful inspection packet filter allows return data.

The filter removes state table entries when the TCP close session negotiation packets are routed through, or after some period of delay, usually a few minutes. This ensures that dropped connections don't leave state table "holes" open. Figure 6.3 shows a filter removing the table entry that allows return data from a connection.

- TCP can only be filtered in the 0th fragments.
- Older packet filters pass all ports above 1024.
- Public Services must be forwarded through the filter.
- Trojan horses can defeat packet filters using NAT.

These hacks are detailed in the following sections.

TCP Can Only Be Filtered in 0th Fragments

Stateless packet filters inspect each packet on its own merits without retaining information about previous packets. For this reason, when a fragmented packet comes through, the TCP header will only be available in the 0th fragment, which means that although the packet filter will drop the 0th fragment, it won't drop fragments past 0.

Because many operating system TCP/IP stacks don't actually pay attention to fragment ordering, these operating systems will reassemble a fragmented packet until they get a packet with the final fragment flag set. If the data in their IP buffer constitutes a valid packet, they'll pass it along to the operating system.

Hackers exploit this to pass data right through a packet filter to a specific host inside your network. By transmitting all packets with the fragment number set to 1, but containing the entire TCP packet, the filter will ignore the TCP payload allowing it to pass to the internal network. The addressed end system sees that the final fragment message is set and passes the TCP packet along to the operating system. In this way, hackers can connect directly to hosts inside your network as if the packet filter didn't even exist.

The two solutions to this problem are to use only hardened TCP/IP stacks on your internal hosts and to use state-based packet filters like true firewalls. Windows NT 4 Service Pack 3 hardens the TCP/IP stack against this vulnerability, but earlier editions of NT are vulnerable. Many Unix implementations are hardened against this attack, but many are not—even some firewalls are vulnerable to this attack. Search the Web for your specific version and variation to determine if this exploit will work against your operating system.

Low Pass Blocking Filters Don't Catch High Port Connections

Stateless packet filters and the packet filtering services provided by ISPs usually open ports above 1024 so that the return socket of a connection can be established back to a host. This means that any protocol running with a TCP port higher than 1024 cannot be protected by these packet filters.

Make sure your packet filter is state-based and blocks all inbound connection attempts except those you want to allow and those connections that were initiated from inside your network.

Public Services Must Be Forwarded

These days, hackers don't usually bother with TCP level exploits because they know you've got to open your web and e-mail servers up to the public anyway. They just use protocol exploits to exploit these servers, and then try to initiate connections to the interior of your network from those servers. If you've set up additional rules in your firewall to allow connections to and from your web server, for example to publish changes to the website, then hackers can then use that access to gain access to the interior of the network.

Internal NATs Can Defeat Filtering

If a user inside your network sets up a Network Address Translator on a machine (a Linux computer, for example), then that machine can be used to perform port and address translation to change a protocol inbound on a high, unfiltered port (say 8080) back down to a protocol on a filtered port (80) and then pass it to an internal server. Internal proxy servers could also be used to cause this vulnerability. This would allow uncontrolled access to your internal services.

Trojan horses perform exactly this sort of protocol translation for the purpose of allowing hackers to access your internal network directly. Unsuspecting users might get e-mail from your e-mail address with instructions to click the setup file of the attached program, which would then install the Trojan horse. Hackers could then exploit the network address translator in the Trojan horse to bypass your filter.

Best Packet Filtering Practices

Pure packet filters are subject to a few glaring security problems. Use these recommendations to keep your packet filtering secure:

- Use at least a stateful filter if you can't use a proxy.
- Disable all ports by default.
- Secure the base operating system.

These practices are detailed in the following sections.

Use a Real Firewall

First and foremost, use a strong firewall. Examine the firewall comparisons that constitute the second half of this book for firewalls that perform correct state-based packet filtering in combination with service-specific, proxy-based filtering and Network Address Translation.

Do not rely upon simple packet filters or the packet filtering functionality of your operating system to keep your network secure. Pure packet filters cannot adequately secure a network.

Disable All Ports By Default

Do not pass all protocols by default and then block those you consider dangerous. Although this is convenient, it opens you up for attack by Trojan horses and unintended user mode services like pcAnywhere.

Block all ports by default, and pass only the ports you intend to serve and the return channels from connections initiated from inside your network.

Secure the Base OS

Make sure the base operating system is secure. As with all security software, the security of a packet filter is based on the security of the device upon which it's run. Most packet filter appliances (routers) can be configured via Telnet. This means that a hacker could telnet into your packet filter and reconfigure it to allow more useful ports to be opened.

Case Study: Leaky Filters

When a customer of mine attached their network to the Internet in 1994, we put a "firewall" in place on the routed connection. In those days, a firewall was a router with a packet filter. We used Telnet to block all inbound ports below 1024 except ports 80 (they ran a web server on a SPARC machine), 21 (FTP), and 25 (SMTP). We felt perfectly secure.

Then a scientist downloaded a Trojan horse embedded in a freeware utility without knowing about it. Since it wasn't a virus, the virus scanning software didn't pick it up.

One day a few months later, the scientist was working at his desk when an MS-DOS command prompt popped up on the screen. At the C:\> prompt, the letters "format c:" slowly appeared one at a time, as if they were being typed by someone directly—except he wasn't typing them.

The hacker apparently paused before hitting the return key for some reason, and during that pause, the scientist wisely unplugged his network connection. He immediately got on the phone to me, and I came in to take a look.

I had never seen such a brazen intrusion before. Until then, Trojan horses to me were a theoretical possibility that were too esoteric to bother thinking about. I knew how difficult it would be to write a useful one, so I didn't worry about them.

This specific Trojan horse set itself up as a server on port 12345 and waited for connections. Upon receiving a connection, it would open a command prompt and vector the I/O to that command prompt over the TCP connection. This effectively gave hackers a remote command prompt on infected machines. Hackers could simply scan wide ranges of IP addresses browsing for open connections on port 12345 to find running instances of the Trojan horse.

Of course, I was called to the mat to explain how the firewall had failed. It hadn't, I explained; it just wasn't designed to prevent that sort of intrusion because this specific firewall could only block ports below 1024, like many early packet filters. The budget for security went up considerably, and we used it to install a Firewall-1-based stateful inspection gateway. We also bought portscanning tools to search for other instances of unknown services running inside the network.

Chapter 7: Network Address Translation

Overview

Network Address Translation (NAT) converts private IP addresses in your private network to globally unique public IP addresses for use on the Internet. Although NAT was originally implemented as a hack to make more IP addresses available to private networks, it has a serendipitous security aspect that has proven at least as important—internal host hiding.

Network Address Translation effectively hides all TCP/IP-level information about your internal hosts from hackers on the Internet by making all your traffic appear to come from a single IP address. NAT also allows you to use any IP address range you want on your internal network (even if those addresses are already in use elsewhere on the Internet, although you won't be able to reach public servers on the public Internet that are within the range of addresses you use on your private Internet). This means you don't have to register a large, expensive block from ARIN or your ISP or reassign network numbers from those you simply plugged in before you connected your network to the Internet.

NAT hides internal IP addresses by converting all internal host addresses to the address of the firewall (or an address responded to by the firewall) as packets are routed through the firewall. The firewall then retransmits the data payload of the internal host from its own address using a translation table to keep track of which sockets on the exterior interface equate to which sockets on the interior interface. To the Internet, all the traffic on your network appears to be coming from one extremely busy computer.

Note RFC 1631 describes Network Address Translation.

NAT is actually a fundamental proxy: a single host makes requests on behalf of all internal hosts, thus hiding their identity from the public network. Windows NT did not provide this function, however Windows 2000 and subsequent Microsoft operating systems can provide Network Address Translation for computers connecting through them to outside networks (and the Internet). Many versions of Unix provide or can use publicly available IP masquerade software. All modern firewalls provide NAT.

NAT is implemented only at the transport layer. This means that information hidden in the data payload of TCP/IP traffic could be transmitted to a higher-level service and used to exploit weaknesses in higher-level traffic or to communicate with a Trojan horse. You'll still have to use a higher-level service like a proxy to prevent higher-level service security breaches.

Tip NAT is so effective at IP address re-use that the implementation of IP version 6 has been practically stalled due to lack of interest, and the threat of IP address scarcity has been eliminated for the foreseeable future. NAT allows an entire class A-sized network to hide behind a single IP address.

NAT Explained

To perform Network Address Translation, firewalls maintain a table of interior sockets matched to exterior sockets. When an interior client establishes a connection to an exterior host, the firewall changes the source socket to one of the firewall's exterior sockets and makes a new entry in the translation table indicating the actual interior source socket, the destination socket, and the mated

firewall socket.

When an exterior host sends data back to the interior hosts socket, the firewall performs the reverse translation. If no entry exists in the translation table for the socket addressed or if the IP address of the source is different than the address the firewall expects to see, then the packet is dropped.

This is easiest to explain with an example. Let's say that interior host 192.168.1.9 wants to establish a web session with exterior host 10.50.23.11. Using the next available port, 192.168.1.9:1234 transmits a TCP packet to 10.50.23.11:80.

Router/Firewall (192.168.1.1 interior address, 10.0.30.2 exterior address) receives the packet, and makes the following record in its translation table:

Source	192.168.1.9:1234
Public Host	10.50.23.11:80
Translation	10.0.30.2:15465

It then transmits the packet on the Internet using the translated IP address and port number, so 10.50.23.11:80 (the public host) receives a connection attempt coming from 10.0.30.2:15465 (the firewall's exterior address). When the public host transmits back, it responds to the source that it thinks originated the request: 10.0.30.2:15465 (the firewall's exterior address).

Upon receiving the packet, the firewall searches its translation table for a matching socket and finds it. It then verifies that the source of the packet is the same as the public host recorded in the translation table when the entry was made. The presence of a table entry confirms that the packet was requested by an internal host—had the packet not been requested, no translation entry would be present matching both the translated socket and the recorded public host socket. If no matching entry is found, the packet is dropped and logged.

The firewall then modifies the packet with the internal source client's socket number and passes it to the interior network for transmission to the ultimate client.

On the public host side, NAT is also being used in "port forwarding" mode—the web server in this case is protected by another NAT, which is configured to receive connections on its public IP address and translate them to the interior of the network. Unlike the NAT on the browser's connection, this configuration is not automatic; the administrator must specifically configure the NAT device for this translation.

In this example, the NAT receives an HTTP connection on 10.50.23.11:80. It examines its port forwarding tables and sees that port 80 is mapped to the interior host 192.168.0.5:80. So the NAT rewrites the IP address from 10.50.23.11:80 to 192.168.0.5:80 and forwards the packet. On the return stream, it performs the inverse translation, so the packet sent to 10.0.30.2:1234 (the browser's NAT's public IP address) from 192.168.0.5:80 is rewritten to come from 10.50.23.11:80 by the NAT device. Figure 7.1 illustrates this process.

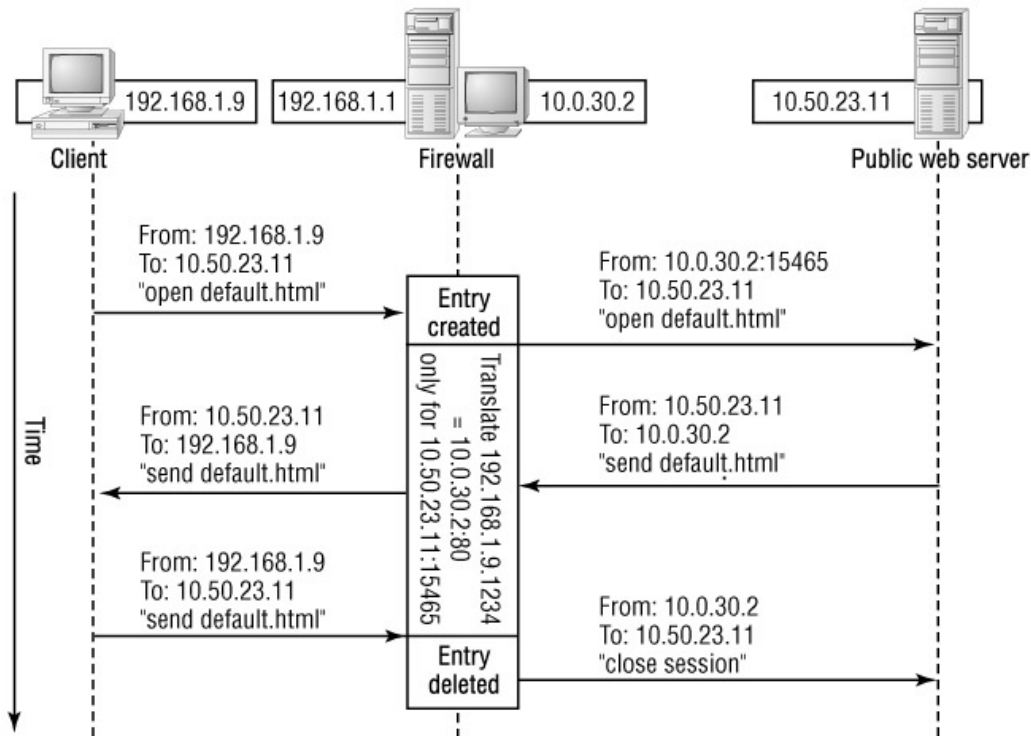


Figure 7.1: Network Address Translation

Because NAT changes the IP address in the packet, it is almost always necessary to make entries in your routing tables to make sure translated packets reach their proper destination inside your network.

In the case of port forwarding, no "dynamic" entry needs to be made or remembered—the IP address rewriting is the same coming and going for every host. Port forwarding is therefore slightly simpler and doesn't require a substantial amount of RAM on the NAT device.

Warning Since NAT performs only simple substitutions at the packet layer, it does not need to perform complex analyses on the contained data, as application proxies must. This means that most implementations of NAT are nearly as fast as straight routing. NAT requires far less processor overhead than higher-level application proxying. Firewalls performing Network Address Translation must have at least one valid public IP address, and that address cannot be concealed.

Since Network Address Translation changes the contents the contents of the IP header, systems that rely on that data remaining unchanged (such as Header Authentication in IPSec, the Internet Protocol Security suite of protocols) will not work through a NAT. Another difficulty with IPSec is that NATs have a difficult time differentiating IPSec traffic coming from multiple interior clients, so firewalls that perform IPSec passthrough typically only allow one interior client at a time to establish IPSec tunnels to exterior locations.

Warning NAT and IPSec don't mix well. Either have your firewall be the end point of your VPN connection (in the ideal situation) or expect only one interior client at a time to use the IPSec passthrough feature of your firewall and kiss header authentication good-bye while you're at it.

Translation Modes

Many firewalls support various types of Network Address Translation. The four primary functions of a NAT firewall are defined below in order of their popularity and availability:

- **Dynamic Translation (also called NAPT, or IP Masquerade)** Wherein a large group of internal clients share a single or small group of internal IP addresses for the purpose of hiding their identities or expanding the internal network address space. Ports on the single public IP address can be forwarded to specified private IP addresses.
- **Static Translation** Wherein a specific block of public addresses are statically translated to a same-sized block of private addresses. In this mode, an internal network resource (usually a server) has a fixed translation that never changes.
- **Load Balancing Translation** Wherein a single IP address and port are translated to a pool of identically configured servers, so that a single public address can be served by a number of servers.
- **Network Redundancy Translation** Wherein multiple Internet connections are attached to a single NAT firewall that it chooses and uses based on bandwidth, congestion, and availability.

Warning Not every firewall supports each type of NAT. Read the firewall's documentation carefully before you purchase it to make sure its form of NAT is the type you need.

Dynamic Translation

Dynamic translation, also referred to as IP Masquerade or just "masking," protects internal hosts by replacing their IP address with an address that routes to the firewall. Individual hosts inside the firewall are identified based on the port number in each connection flowing through the firewall.

Note RFC 1631 does not describe the use of port addresses to extend the applicability of a single IP address, but every existing NAT implementation I know of uses this method. Purists call this method NAPT, for Network Address and Port Translation.

Because a translation entry does not exist until an interior client establishes a connection out through the firewall, external computers have no method to address an internal host that is protected using a dynamically translated IP address. And since most firewalls create translations that are valid only for the addressed host and port, there's no way for any computer except the computer addressed to attack the host because no other route exists back to it.

Technically, it is possible to use the Internet Protocol's source-routing feature to route through a NAT. Source routing allows you to specify intermediate routers through which a packet must travel. By specifying the NAT device as an intermediate router between a public machine and an interior private address, it is possible to route packets through a NAT. But because any NAT sold as a security device is configured to drop any packets that are source-routed, this can only be accomplished through simple or improperly configured NAT devices. You should test any NAT devices you deploy to ensure that they do drop source-routed packets for this reason.

It's important to note that NAT does nothing to protect the client other than to keep external hosts from connecting to it. If the client is seduced into connecting to a malicious external host, or if a Trojan horse is somehow installed on the computer that connects to a specific external host, the client can be compromised just as easily as if there were no firewall. For this reason, NAT alone is not sufficient to protect your network.

Seducing a client into connecting to a malicious site is surprisingly easy. For example, if your boss

sent you e-mail saying, "Check this site out. It's remarkably close to what we want to do," you'd probably click on the hyperlink included in the e-mail without a second thought. That's all it takes, and forging e-mail is child's play for a hacker.

Warning The paragraph above was originally written just days before the Explore.Zip worm wreaked havoc upon corporate networks using exactly that strategy. You should strongly consider filtering executable files from HTTP down-loads and e-mail attachments.

Reality Check: Forging E-Mail to Gain Control

Forging e-mail to gain control of a computer is easy if the intended victim views their e-mail in HTML format and has JavaScript enabled (which is the default configuration for both Outlook and Outlook Express). In this case, the e-mail can contain Java-Script event triggers that will automatically start your web browser and pull up the page of the attacker's choice, which would subsequently allow the malicious website operator to perform every heinous act listed on Microsoft's support site when you search on "malicious web site operator."

Worse, if you've ever selected the option to automatically open executable down-loads and the attacker points your web browser at an executable file, your web browser will automatically download it and then execute it, all without asking you anything. To show how simple this exploit is to perpetrate, import the following HTML/JavaScript code into an Outlook Express e-mail using the import text feature (the import is a bit tricky and doesn't just work by cutting and pasting), then send the e-mail to yourself. If you have JavaScript enabled and view your e-mail as HTML, the Outlook Express will open Google's search page automatically when you close the e-mail message.

```
<html>
<head>
<title>This is funny!</title>
</head>
<body onunload="Leave()">
<script language="JavaScript">
var leave=true;
function Leave()
{if (leave)
open("http://google.com");}
</script>
Hi Folks, This is funny.<p>
</Body>
</html>
```

Some protocols do not function correctly when the port is changed. These protocols will not work through a dynamically translated connection. Any protocol that relies upon the ability to establish a separate reverse connection to the source client will only work correctly if the firewall is designed to make exceptions for that specific protocol.

When you use dynamic translation, you must establish an IP address to translate the internal addresses to. This is the address that will be visible to the outside world for outbound connections. Most firewalls allow you to use the firewall's own address or another address that routes to the firewall and for which the firewall will answer using ARP.

Each IP address can only support a theoretical maximum of 65,536 (or 2^{16}) connections because the port address pool used for multiplexing the client connections is only 16 bits wide. Most firewalls are further limited to about 50,000 connections because many ports are reserved for other uses.

Linux's default IP Masquerade settings make only 4096 ports available for translation, but that number can be easily modified.

In any case, the number of ports is large and shouldn't cause a problem unless your users maintain hundreds of simultaneous Internet connections while they work. If you do find yourself running out of ports, you'll have to have more than one IP address behind which to hide hosts.

Static Translation

Static translation is used when you have resources inside your firewall that you want to be publicly available or (in rare cases) when you use a protocol that must have certain port or IP addresses to operate.

Static translation can be used to map a range of public IP addresses to the same-sized block of internal private addresses. For example, you could translate 128.110.121.0–128.110.121.255 to the internal range 10.1.2.0–10.1.2.255. The firewall performs a simple static translation for each of the IP addresses in the range.

Port forwarding is a type of static translation that refers to the process of forwarding just a specific port, rather than an entire IP address or block of addresses. Let's say your e-mail server's IP address is 10.1.1.21, and your firewall's external IP address is 10.0.30.2. You can statically map socket 10.0.30.2:25 to address 10.1.1.21:25. This static connection will cause the firewall to translate any connections to its SMTP port to the e-mail server inside your firewall.

Port forwarding can be used to establish a number of different complex services on a single IP address. For example, you could have an e-mail server statically translated on the SMTP and POP ports, a web server statically translated on the HTTP port, and a news server on the NNTP port. Since the translations can specify any IP address, these services can be split among many machines inside your firewall.

Load Balancing

Some firewalls support IP load balancing using the static NAT facility. This allows you to spread the load of one very popular website across a number of different servers by using the firewall to choose which internal server each external client should connect to on either a round-robin or balanced load basis. This is somewhat similar to dynamic translation in reverse—the firewall chooses which server from among a pool of clones each connection attempt should go to.

To choose based on load, the servers in the pool must have some facility to transmit their load levels to the firewall. Since there is no standard way to do this, your firewall must implement a proprietary method. For that reason, many simpler firewalls assume that each connection creates about the same amount of load and assign connections to the next server in the list. Other more sophisticated load-balancing firewalls attempt to average the number of simultaneous connections to each interior host or attempt to average the amount of network bandwidth exchanged with each interior host.

IP load balancing only works with protocols that are stateless or maintain their state on the client. For websites, IP load balancing is perfect because the server does not maintain any information about the client between page transmittals, so it doesn't actually matter if a specific client gets the same server each time they load a page. Consider the problem with mail, though. If a firewall provides load balancing for a number of e-mail servers, each of a user's e-mail messages would arrive on any one of the servers depending on which server the firewall selected for the SMTP

connection. When the user connected to a server, the firewall would again select one server for the POP connection so the only messages that user would see are the messages that happen to have been received by that server—the user would not see all of her received messages.

IP load balancing is particularly important for e-commerce sites that have a heavy processing load because they make heavy use of Active Server Pages, CGI or Perl scripts, or Java servlets. These technologies all put a heavy compute burden on a web server, which reduces the maximum number of clients that server can support. Figure 7.2 shows a complex e-commerce website being load balanced by a firewall.

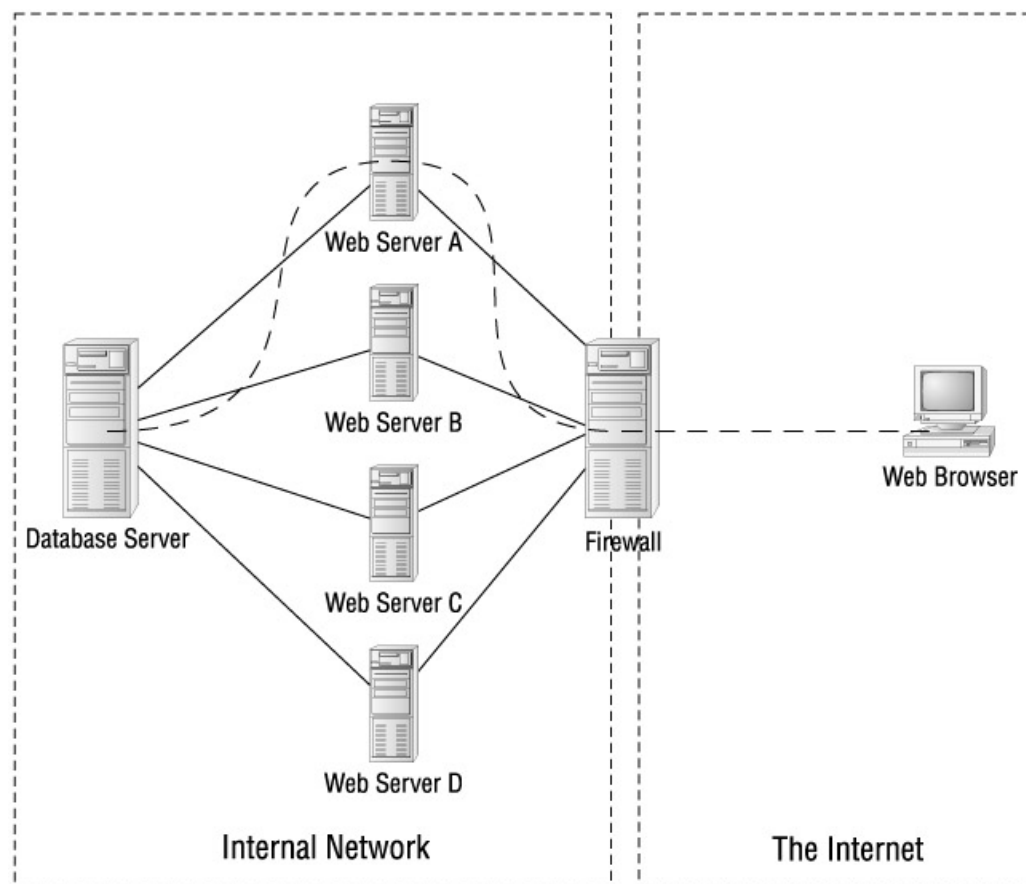


Figure 7.2: Using firewalls to perform load balancing
Network Redundancy

It is possible to use a NAT-based firewall to perform Internet network redundancy either to balance the load of clients across multiple low-cost Internet connections or to compensate automatically for the failure of any given link.

Network redundancy works with dynamic translation in much the same way that IP load balancing works with static translation. In network redundancy, the firewall is connected to multiple ISPs through multiple interfaces, and has a public masquerade address for each ISP. Each time an internal host makes a connection through the firewall, the firewall decides on a least-loaded basis which network to establish the translated connection on. In this way, the firewall is able to spread the internal client load across multiple networks. The failure of any network is then treated as if that network is completely loaded; the firewall simply will not route new clients through it. Although session-based protocols will have to be reestablished from the client hosts, stateless protocols like HTTP could then survive a link failure without the client even knowing anything had occurred. Figure 7.3 shows network redundancy. Notice that because the IP address is translated, it does not matter

which ISP the firewall uses to connect to the public website.

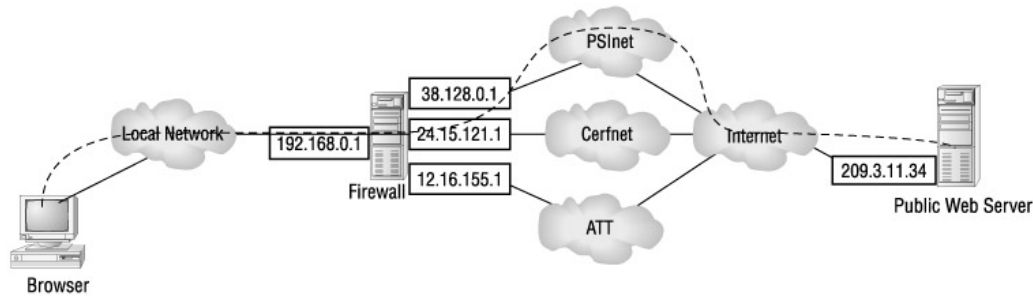


Figure 7.3: Network redundancy

Router Configuration for NAT

When you use Network Address Translation with IP addresses other than the IP address of the firewall, you'll have to configure the routing in your network to make sure that packets reach the firewall, and you may have to configure routing on the firewall to make sure that packets are relayed to the correct interfaces.

Whether or not you have to configure routing separately from the firewall's configuration depends upon whether the firewall routes packets itself or whether it relies upon the host system for routing. If the firewall relies on the host operating system to perform the routing function, you then need to know whether the firewall translates addresses before or after the routing function has occurred.

You can tell whether or not the firewall relies upon the host system for routing in Unix by checking whether the firewall requires the use of the `routed` daemon. If the firewall requires the `routed` daemon, then it relies upon the `routed` daemon to perform the routing function.

In Windows, you can tell if the firewall relies upon the operating system to route if the firewall enables the `Enable IP Forwarding` setting in the network Control Panel, or if it instructs you to enable it manually.

If the firewall relies on the operating system for routing, you must ensure that the internal routing tables are correct for the various translation modes you establish. Some firewalls configure the routing tables for you; others do not. If the firewall performs the routing function, you can assume that the firewall will correctly route packets without intervention. In either case, you should thoroughly read the firewall documentation on routing and test the routing through your firewall once it's completely configured.

The first router between your firewall and your ISP is usually the biggest routing problem, because it may assume that it can use ARP (Address Resolution Protocol) to determine where a packet should be routed, especially if there's no static route between the router and your firewall. This means that your firewall will have to respond to ARP requests for all the IP addresses that you want to pass through the firewall.

Most true routers will allow you to manually set a route to the correct interface on the firewall, so this isn't an issue. But many new high-speed data link devices like cable modems and DSL interfaces aren't actually routers—they're just bridges, and they assume that every device they talk to will be connected to the same collision domain as their Ethernet adapter.

In Unix, this is no problem. Just use the `ARP` command to set the IP addresses for which the external interface will use ARP (or respond as the correct interface for said IP address), and add routing entries to make sure the packets will be routed to their correct location as shown in Table

7.1.

Table 7.1: Firewall Routing Table Entries

Network	Mask	Gateway
10.0.0.0	10.0.0.255	10.0.0.1
128.110.121.0	128.110.121.255	10.0.30.2

In Windows NT, this is a serious problem. Windows NT includes an ARP command, but it does not properly implement the ARP protocol for proxy ARP (when an interface responds to ARP requests for multiple IP addresses). Manual ARP entries remain cached for only about an hour, after which time your routing will fail.

The only way to handle this problem is for the firewall software to implement some method to properly implement proxy ARP for the IP addresses that the firewall uses for translation.

Adding IP addresses to the external adapter will not solve the problem because these addresses are automatically used to create the routing tables for NT, which will then be incorrect for further routing to the interior of the network.

An example will explain why this is the case. If the firewall has two interfaces, 10.0.0.1 and 10.0.30.2, then there will be default entries in the routing table that look like Table 7.1.

Say host 10.0.0.12 has its address translated to 128.110.121.44 as it goes through firewall 10.0.30.2. A return packet will be addressed to 128.110.121.44. When that packet reaches the firewall router, the packet will be routed to interface 10.0.30.2 by default because the address translation will have occurred after the routing function had already happened. The firewall will then translate the address to 10.0.0.12, but it will have been too late—the packet will have already been routed to the incorrect interface so it will be transmitted on the 128.110.121.0 network.

If you try to solve the problem by adding IP address 128.110.121.44 to the 128.110.121.1 interface or the 10.0.0.1 interface, Windows NT will automatically generate routing rules for that interface that unfortunately stipulate incorrect routing—and these automatically generated rules cannot be removed.

To solve the problem, use the firewall's ARP facility and make a routing entry on the firewall that specifies a route for each IP address that the firewall proxy ARPs:

```
Route add 128.110.121.44 10.0.0.1 1
```

This will ensure that packets coming in on the proxy ARP address will be routed to interface 10.0.0.1. Then they will be transmitted on the correct interface once the address is translated.

IANA Private Use Network Numbers

The Internet Assigned Numbers Authority (IANA) has designated three blocks of addresses for private use without coordination:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

Note The IANA function is now operated by the Internet Corporation for Assigned Names and Numbers (ICANN). The term now officially refers to the function provided by the former IANA organization, as now operated by ICANN.

Internet routers are configured not to route these addresses on the Internet backbone. You can use these addresses in your own network with a certain amount of impunity, unless your ISP also uses them. Most ISPs use portions of the 10.0.0.0 domain for their own internal routing, with Network Address Translation into and out of the range.

For that reason, I generally recommend that clients use the 192.168.0.0 network range for their own private networking. This prevents conflicts with ISPs that use the 10 domain for internal routing, and you don't have to remember which block of addresses is valid inside the 192.168.0.0 domain. You can use all 16 bits of address space with impunity.

Problems with NAT

There are a few protocols that cannot be used with NAT because they either require the ability to open a back channel to the client, embed TCP/IP address information inside the higher-level protocol, encrypt TCP header information, or use the original IP address for some security purpose.

- Back channels will not work because no separate route back to the internal hosts exists. This occurs with H.323 video conferencing.
- Software that embeds TCP/IP address information inside TCP/IP packets and then relies upon that information will not work because the interior TCP/IP address information will be incorrect. This occurs with FTP and some other protocols.
- Software that encrypts the TCP header information will not work correctly with NAT because the TCP information must be accessible to the firewall. Solve these problems by making the firewall the encryption end point. This occurs with PPTP and IPSec Header Authentication.
- Software that relies upon TCP/IP address information for security checking will fail because the IP address information has changed. This occurs with Sqlnet2.

More advanced firewall software can inspect outgoing connections for these protocols and establish a translation entry to wait for the destination public host to respond with the back-channel open request. Most firewalls do not support service-specific NAT; rather, they use service-specific proxy software in combination with the NAT mechanism to perform these functions.

NAT cannot be used with the following services without some form of higher-level proxying or a patch to the basic NAT code:

- **H.323, CUSeeMe, and VDO Live** These video conferencing software programs cannot be used because they rely upon the ability to establish a back channel to the host. Some firewalls may make special entries in their translation tables to allow a specific host to create the back channel.
- **Xing** This software fails for the same reason that video conferencing programs do.
- **Rshell** This software fails for the same reason that video conferencing programs do.
- **IRC** This software fails for the same reason that video conferencing programs do.
- **PPTP** This software fails because it relies on encrypted IP information inside its stream, but other non-TCP/IP protocols can be tunneled inside PPTP with an end point on the firewall to get around this problem.
- **Sqlnet2** This software requires the numerical difference between the host and client IP addresses to be the same as if both IP addresses were not translated. This means that the protocol will nearly always fail unless you design your network around this strange

restriction.

- **FTP** This software must be RFC 1631–compliant to work with NAT. FTP embeds IP address information in ASCII text inside the TCP packets, the length of which is likely to change through an address translation.
- **ICMP** This software sometimes embeds the first part of the original packet in ICMP messages. This first portion will contain the untranslated address. The secure solution is simply not to pass ICMP traffic through your firewall.
- **IPSec** IPSec–authenticated headers cannot pass through a NAT because AH uses a hash to determine if any part of the packet has been changed in transit. The purpose of NAT is to change the IP address, so the AH hash will no longer be correct and the end systems will drop the AH packets because they fail authentication.

ESP in transport mode cannot transit a NAT for the same reason that AH can't—checksums won't match if the IP addresses have changed. However, some VPN clients allow the administrator to ignore checksums, so these implementations may be able to get ESP transport mode through a NAT.

ESP in tunnel mode (wherein the original packet is encrypted and becomes the payload of a non–encrypted packet) can traverse static NAT mappings. However, when using dynamic (NAPT) translation, there's no way to differentiate which interior host different inbound IPSec connections should go to, so only one IPSec connection will work through the dynamic NAT.

Finally, most implementations of IKE use pre–shared keys to prove their identity and bind the original IP address as part of the key. Again, since the original IP addresses change, the key changes, and the IKE negotiation fails. The solution in this case is to configure your firewall and VPN clients to use X.509 certificates or some proprietary keying mechanism to transit the NAT.

Basically, configuring IPSec to transit a NAT is much harder than simply end–pointing the VPN on the firewall. That's the logical and most secure place to do it anyway.

Hacking through NAT

If Network Address Translation makes clients invisible, it's impossible to hack them, right? Wrong. Here's where NAT can fail:

- Static translation does not protect the internal host.
- If the client establishes the connection, a return connection exists.
- If the connection can be intercepted or subjected to a man–in–the–middle attack, the man–in–the–middle is the end point.
- If the firewall's NAT implementation is flawed, it could be subject to exploitation.
- Source routing can specify the NAT device as an intermediate system and then route through it.

These hacks are further explained in the following sections.

Static Translation = No Security

Static translation merely replaces port information on a one–to–one basis. This affords no protection to statically translated hosts—hacking attacks will be just as efficiently translated as any valid connection attempt.

The solution to security for statically translated hosts that must provide public services is to reduce the number of attack vectors to one, and then to use (if necessary) application proxy software or other application-based security measures to further protect the internal host.

For example, to protect e-mail servers inside a firewall, Checkpoint's Firewall-1 software has a simple SMTP store-and-forward mechanism that eliminates the nasty problem of e-mail buffer overflows. Firewall-1's e-mail proxy receives all e-mail inbound on its port 25 (SMTP) and writes each message to the hard disk. A separate service running in a different process reads the e-mail messages and transmits them to the internal e-mail server. This mechanism eliminates the possibility that a buffer overflow from a malformed e-mail message will affect anything but the firewall, and in that case, it will only shut down the e-mail receiver. The firewall's internal consistency checker can then simply restart the e-mail receiver afresh.

Internal Host Seduction

Even if hackers can't get inside your network, you can't prevent your users from going to the hackers. Forged e-mail with a website link, a Trojan horse, or a seductive content website can entice your users to attach to a machine whose purpose is to glean information about your network. HTTP is a reasonably robust protocol, and extensions to it by the major browser vendors have made it a potent source for exploitation. Since nearly any type of content can be downloaded through an HTTP connection, your users can be easily compromised once they connect to a hacker's machine.

Higher-level, application-specific proxies are once again the solution. By inspecting the contents of those protocols for what will most likely be exploited (like HTTP), your firewall can sniff for suspicious content like Java applets, ActiveX controls, downloadable executables, malformed URLs, and other tools hackers use to perform their mischief. By filtering out or at least logging and alerting on the presence of this content, you can keep your users from being exploited through seductive means.

The State Table Timeout Problem

Network Address Translation has one hackable aspect to its function. When a client connects to the outside world, the firewall must remember which internal host has connected to which external host so that it can route the return portion of the TCP connection back to the source host. Without this functionality, only unidirectional (and therefore worthless) transmissions can be made.

But how does the firewall know when the client is finished with the connection? If the firewall relies only upon TCP information, it can sniff for close session information. But many protocols don't have an obvious ending negotiation, and there's always the possibility that a connection may be dropped without proper closing negotiation. In those cases, to remain as protocol generic as possible, most firewalls implement a timeout value. The length of this timeout varies greatly, and firewall publishers never tell you how long it is.

Before the timeout occurs, a connection through the firewall to the originating internal host exists. This connection could potentially be exploited by a hacker, although the hacker would have to know exactly which IP address and port to use, and would probably have to know the IP address of the original destination in order to get through the firewall. This information would only be known if the hacker had been snooping on the connection.

The theoretical possibility of this exploitation exists, but would require a lot of information to execute. Even if the client had not closed the connection, it wouldn't be listening on the high number port it

had opened to initiate the connection. This attack could be most effectively used to send malformed packets to a host in a denial-of-service attack.

Source Routing through NAT

Dynamic NAT naturally protects clients by eliminating a direct route to them from the Internet. Since no forward route exists, there's simply no way to get to an internal client from the Internet—any attack stops at the NAT device.

Unless the hacker uses source routing and knows an internal IP address. Source routing is a feature of IP that was originally used for testing, but is now most often exploited by hackers to hijack TCP connections by specifying routes back to themselves rather than to legitimate hosts.

In a normal IP packet transmission, the header contains just the source and destination addresses, and relies upon a series of default gateways and routers to know how to get it from one router to the next.

With source routing, the source host of the packet can specify a list of routers that the packet must be transmitted to other than the default route. The mechanism is simple—rather than transmitting the packet to the final destination, the packet is forwarded to the next router in the source router list. Once it arrives at that router, the router sends it to the next address in line—either another router or the final system. There are two flavors of source routing: *strict*, where the entire router list is specified and the packets will be dropped if they can't make it using that route, or *loose*, where one or more intermediate systems are listed, but where any number of other intermediate systems can be transited between the listed systems.

So, to get through NAT, the hacker need only specify the final internal IP address of the host but include the NAT as an intermediate system in a loose source-routed packet. The original host will then transmit the packet to the NAT device, which will then transmit the packet to the interior of the network because it does know the identity of the final destination.

Of course, the hacker would have to know the network numbering scheme inside your network. But that's easy. By simply scanning the standard private ranges (10, 192.168., etc.) using source-routed ICMP packets, he'll see ping responses from whatever computers exist and be able to quickly build a map of the internal portion of your network.

The solution is to configure your firewall to drop source-routed packets, since the few legitimate purposes for them are easily accomplished using other means and since no standard operating system or router generates them by default.

Warning Be sure you test simple NAT devices for the source-routing vulnerability before you deploy them as security on home or small office networks.

Case Study: Trial by Firewall

A few years ago, a customer of mine wanted to implement Network Address Translation in their network for security reasons and because they used a DSL adapter that only provided 16 public IP addresses. Since they were an all-Microsoft shop and a small business, we decided to use Checkpoint Firewall-1, which has good automatic support for Network Address Translation.

After setting up the firewall for standard use and creating the appropriate policies, I set up Network Address Translation for the internal client computers to use the firewall's IP address. I then created

a static translation for the mail server, which was internal.

The client access worked fine, but I realized that I'd have to set a route in the DSL adapter to route the static IP address for the mail server. Since the DSL adapter was owned by the ISP, I called their technical support number and asked to have a route programmed into the adapter.

Much to my surprise, the support technician had no idea what I was talking about. He'd never heard of adding a route to anything. I asked to speak with his supervisor, who then informed me that the DSL adapter was not a router at all—it was merely a Data Link–layer bridge. Any devices it talked to would have to be present on the same Ethernet collision domain.

This was problematic because we needed to protect the e–mail server (and because I didn't want to buy a hub to put between the firewall and the DSL adapter). So I decided to use proxy ARP, which I was familiar with from the Unix world, but which I'd never used in Windows NT.

I sat down to the NT server and typed ARP. The command existed and it worked as expected. I typed in the ARP command to bind the static translation address to the MAC address of the external adapter. It accepted the command, and viola! Traffic began to flow. I verified that I could connect from the outside world to the e–mail server and that mail could flow from the ISP. Everything seemed fine.

After chatting with my contact at the client site, I packed up my laptop and tools and began to leave. At that point, one of the employees said that the e–mail service seemed to have gone down. I checked for connectivity again, and found that indeed the static mapping no longer worked.

I was stumped. I checked the firewall—no difference. I re–issued the ARP command with the parameter to indicate a static mapping. That worked as it had the first time, and traffic flowed again.

This time I was wary. I left my laptop in place running a constant ping from the Internet (through a dial–up account) to the mail server's static address mapping. Sure enough, about an hour later the ping began to fail.

For some reason, Windows NT would not retain the ARP mapping. I fired up Internet Explorer and searched Microsoft's tech support site to no avail. Almost no information existed on the default ARP command.

So I went to Alta Vista and did a net–wide search. Surprisingly few pages matched, but I did find one that specifically mentioned that NT's ARP command could not retain address resolutions—exactly the problem I was having.

I figured Firewall–1 must have some sort of solution to the problem. I pulled up the documentation on the CD–ROM (which is normally very complete) and dug around. All I found was a description of the proxy ARP problem with the solution that you should manually enter a proxy ARP command to solve it, which I'd already done.

Next, I added the statically mapped IP address to the protocol stack of the NT machine as a second IP address on the external adapter. Unfortunately, Firewall–1 performs late translation, which means that the IP addresses are translated after the operating system has routed the packet. This means that the packets bound for the statically translated address were being "bounced" back out the external adapter by the default routing NT adds for each adapter, which I could not defeat. Only true proxy ARP, which NT didn't support, would solve my problem.

I couldn't imagine that I was the first person to have had this problem, so I called Checkpoint's tech support. They told me they could open up a support incident for \$500, which frosted me, but which I would gladly pay if they could guarantee me a solution. They would not, so I decided not to pay them.

I sat and stewed about the problem when my phone suddenly rang. It was the Checkpoint support technician, who said he'd send me e-mail with a URL to click on. I checked my mail, read the message, and clicked into a private area in Checkpoint's support website that described the exact problem and how to solve it.

It seems that Firewall-1's firewalling engine for NT is designed to support proxy ARP because NT doesn't, but that there's no user interface for doing it or documentation on how to do it. Firewall-1 will ARP by adding entries to a file in Firewall-1's configuration directory called config.arp. The file is a simple list of Ethernet MAC addresses for the adapter and the IP address to respond to.

I added the entry for the static mapping and restarted the firewall. Everything worked fine after that.

This problem highlights my basic problem with the fee-for-support arrangements that nearly all firewall vendors set up. Vendors are naturally inclined to "hold out" on users until they pay for support, and to provide minimal documentation so as to increase the number of paid support calls. Vendors claim that they have to pay their support costs, but making user interfaces and completely documenting software can nearly eliminate support calls. Be sure to do a thorough search on the Internet and at Google Groups (groups.google.com) on your issue before paying for technical support.

Chapter 8: Application–Level Proxies

Overview

Proxy servers were originally developed to cache frequently accessed web pages for computers behind a common Internet connection. In the early days of the Internet, wide area links were very slow, the Web was relatively small, and web pages were static. The entire Web consisted of only a few thousand websites shared by scientists and academicians. Whenever an important news element hit a website, many scientists in the same organization would visit that page (how many times have you forwarded a link inside your company?). By caching that page on a local server, proxies could eliminate redundant Internet access to retrieve the same page over and over. So, proxies were originally very effective at web caching.

When the Web went supernova, proxies became markedly less effective at caching; the Web was now vast, web pages were frequently dynamic (expiring as soon as they'd been transmitted), and the interests of users within a single organization might range across a million web pages before the same site was hit three times. These factors presented a difficult caching problem indeed and proxies became largely ineffective, except in extremely large organizations or in ISPs. Although support for proxy servers was built into all the standard browsers, by 1996 it was seldom used.

But the new Web also has its seedier element, and proxy servers showed a remarkably serendipitous side effect: They can hide all the real users of a network behind a single machine, they can filter URLs, and they can drop suspicious or illegal content. So although originally created as non–security caches, the primary purpose of the majority of proxy servers has now become firewalling.

Proxy servers regenerate high–level service requests on an external network on behalf of their clients on a private network. This effectively hides the identity and number of clients on the internal network from examination by the external network. Because of their position between a number of internal clients and public servers, proxies can also cache frequently accessed content from the public network to reduce access to the public network through high–cost wide–area links.

For the sake of understanding, this chapter discusses only "pure" proxies—those that operate on the principle of service protocol forwarding. Most actual implementations of security proxies include the services of packet filtering and Network Address Translation to form a complete firewall. Those technologies can be combined with proxies to eliminate some of the attacks to which pure proxies are vulnerable.

Many proxy service alternatives exist, ranging from the Application layer filter functionality of true firewalls like Checkpoint's Firewall–1, to general–purpose pure "proxy only" applications like WinGate, to simple single–service proxies like Jigsaw for HTTP. Pure proxies are subject to a number of problems, most based on the fact that the base operating system is not protected by the proxy software against denial–of–service attacks and the exploitation of other services that may be running on the server.

Proxy servers are most often associated with the HTTP World Wide Web service because proxies were first developed for this service. Since that time, proxy functionality has been applied to most other common Internet services. Examples in this chapter will use the HTTP service, but the functionality remains largely the same for other services.

How Proxies Work

Proxies work by listening for service requests from internal clients and then sending those requests on the external network as if the proxy server itself were the originating client. When the proxy server receives a response from the public server, it returns that response to the original internal client as if it were the originating public server. Figure 8.1 shows this process in detail.

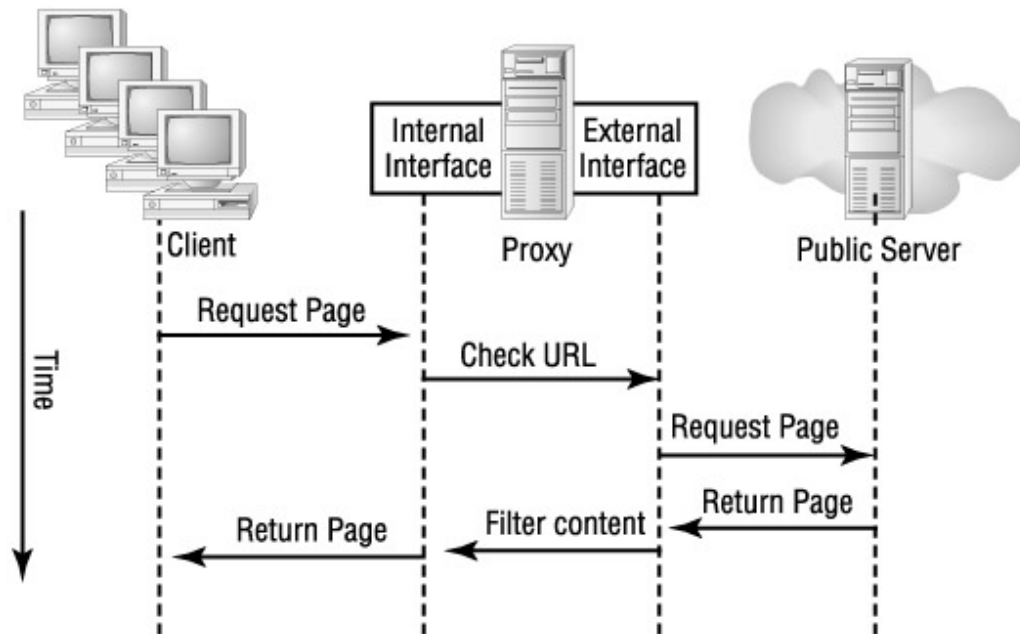


Figure 8.1: A service proxy

The next sections discuss the advantages and disadvantages of proxy servers.

Security Advantages of Proxies

The process of request regeneration and the fact of a proxy's location between the external and internal networks provide a number of security advantages:

- Proxies hide private clients from external exposure.
- Proxies can block dangerous URLs.
- Proxies can filter dangerous content such as viruses and Trojan horses before passing it to the client.
- Proxies can check the consistency of returned content.
- Proxies can eliminate the need for transport layer routing between networks.
- Proxies provide a single point of access, control, and logging.

Each of these security advantages is detailed in the following sections.

Client Hiding

The major security feature of proxy servers is client hiding. Like Network Address Translation, proxy servers can make an entire internal network appear to be a single machine from the Internet because only a single machine passes requests onto the Internet.

Like Network Address Translators, proxy servers prevent external hosts from connecting to services on internal machines. In the case of proxy servers, no route to the clients exists because the address domains of the internal and external networks may be incompatible and because transport

layer routing does not exist between the two networks.

Proxies perform this feature by completely regenerating service-level requests rather than simply changing and recalculating address headers. For example, when a web client makes a request through a proxy server, the proxy server receives the request as if it were the destination web server on the internal network. It then regenerates the request on the external network as if it were a typical web browser. When the proxy receives the response from the ultimate web server, it serves that response to its internal client. Only HTTP passes through the proxy, not TCP or IP. TCP/IP (and other low-level protocols) are regenerated by the proxy; they do not route through it unless the proxy is misconfigured.

Another aspect of client hiding is that of connection multiplexing; a proxy server can be used to share a single Internet connection and IP address among an entire network. For this reason, light proxy servers like WinGate are extremely popular in home and small office environments where only a single dial-up or dedicated connection is available.

URL Blocking

URL blocking allows administrators to disallow the provision of certain websites based on their URLs. In theory, this will keep your employees from viewing websites you don't want them to have access to. This function is easy to implement. The proxy simply checks every request for a web page (or other service URL) against a list of denied pages before it regenerates the request. If the URL is blocked, the proxy will not request or return the page.

URL blocking is easy to circumvent, however, because a website can be just as easily addressed by its IP address or even by the whole number address. For example, a user could type in any of the following in their web browser to access exactly the same home page:

```
http://www.gamehound.com/default.html  
http://192.168.13.12/default.html  
http://3232238860/default.html
```

But your URL blocker will (probably) only be checking for the full text URL.

URLs can contain DNS names or IP addresses. Most people are familiar with the first two examples of site references, but have never heard of the third: an IP address specified as a whole number rather than as a "dotted quad notation." The concept is simple: An IP address is just a 32-bit number, and though we refer to them in dotted quad (10.0.0.0) notation for convenience sake, there's no reason why they can't be referred to as whole numbers. To convert a dotted quad number to a whole number, use the following formula ("a" is the most significant quad, "d" the least): $a \times 2^{24} + b \times 2^{16} + c \times 2^8 + d$. Converting everything to easily calculable numbers, the formula becomes: $a \times 16777216 + b \times 65536 + c \times 256 + d$. So, for example, turning the IP address for <http://www.starlingtech.com/>, 209.68.11.152, into a whole number makes it $209 \times 16777216 + 68 \times 65536 + 11 \times 256 + 152 = 3510897560$. Put 3510897560 into your web browser's address bar and you'll see the Starling Technologies home page come up. Note that websites behind proxy servers (like Microsoft.com) don't come up because the whole number IP address must be programmed into the proxy for the proxy to recognize it.

The other major problem with URL blocking for security administrators is simply keeping up with sites to block. Problem sites like hacking depositories, pornographic sites, and game sites have the ephemeral life of a mayfly—they pop up and disappear just as quickly. Most people who engage in the activities ascribed by these sites just use search engines or Usenet news lists to keep up with where their favorite sites have moved. You will not be able to stay ahead of that activity with your

Sage Advice: Don't Make Me URL

When you're dealing with adults, it's much easier to use policy to enforce appropriate web use. Just make sure everyone knows that every website accessed is cached on your proxy server, and that the proxy uses content filtering to search for suspicious content like games, pornography, and the city of San Francisco in website content. Once they know that you could check up on their browsing after the fact, they won't risk it, and you won't have to deal with trying to stay ahead of a blocked URL database.

(Disclaimer: The San Francisco joke is a reference to the fact that Sybex, the publisher of this book, is located in the San Francisco Bay Area. It should not be construed as an endorsement or denigration of vegetarianism or any other lifestyle choice.)

Content Filtering

Because proxies retransmit all protocol payloads and are protocol specific, the proxy service can be used to search the payload for suspicious content. This means that you can configure your HTTP proxy service to strip out ActiveX controls, Java applets, or even large images if you feel they could present a security problem. You could also use an SMTP proxy to strip out executable file attachments and archived zip files if you felt they were a security problem.

Content filters can also be used to check web pages for the presence of certain words or phrases, such as the trademarks of your competition or some current news item.

You should filter ActiveX controls in websites, Java applets, and executable files in e-mail because they can be used to install Trojan horses inside your network. If someone needs to transfer an executable file, have him or her transmit it as a zip file or use BinHex or some other encoder to transfer it in a text format. This will require effort to decode, thus preventing the accidental transfer of a virus or Trojan horse into your network.

Consistency Checking

Consistency checking refers to checking the content of a protocol to be sure it makes sense for that protocol. Consistency checking ensures that specifically malformed types of content can't be used to exploit a security weakness in your internal network.

For example, earlier versions of the Unix Sendmail SMTP daemon were notoriously susceptible to various buffer overflow problems. These occurred when an e-mail message was sent and it was longer than it said it was. Sendmail would allocate a memory buffer that was the size the message claimed to be, but would then scan the message until it hit an end-of-file marker. If the area between the claimed end and the actual end contained executable code, a hacker could gain root access to your e-mail server.

Sage Advice: Beware the Open Filter

I have customers who use content filtering to keep employees from reading news sites, posting their resumes on job sites, and basically doing anything even remotely fun on the Web. The problem they've run into is that the employees now feel that any site that isn't blocked is fair game—if they

can get to it, they can waste as much time as they want on it. The whole thing has turned into a sort of escalating cold war between the IT department and the rest of the company. Once an unblocked site is found, it's passed around discreetly (and not through e-mail) amongst the employees until everyone knows about it.

Again, my earlier advice of simply treating people like adults and making sure they know that you could check up on their activities if you wanted to is a far more effective method than blocking or filtering specific types of content.

Another example of a buffer overflow occurs in URLs that are longer than 256 characters. Early web browsers were flawed because the end of the URL beyond 256 characters could contain executable code that would be executed by the browser software.

Of course by now everyone has heard of the amazing number of buffer overrun exploits that hackers have been able to perpetrate against IIS4 and IIS5. Using URLs and posts that are longer than IIS can handle, as well as exploits against supporting DLLs like the text search and legacy database support modules of IIS, hackers have been able to create completely automated attacks against IIS that operate as worms on the Internet and cause widespread havoc. Microsoft has released hundreds of security hotfixes to try to cope with the problem, and it has seriously impacted the marketability of their .NET services, which are all based on a web server that nobody wants to deploy because of its security problems.

Consistency checking with your proxy software can ensure that these sorts of problems are eliminated at the proxy so they won't affect internal machines. Unfortunately, the problems to check for usually are not known until some hacker exploits them, so most consistency checks are only available after an exploit has been found. And with automated worms, a large portion of the web servers on the net can be exploited within a few hours, so the "countermeasure" aspect of hotfixing servers is rather ineffective.

Route Blocking

Transport layer packets need not be routed because the request is completely regenerated. This eliminates Transport layer exploits like source routing, fragmentation, and various denial-of-service attacks. By eliminating routing, you can also ensure that any protocol for which you have not established a proxy service cannot be passed to the public network.

Route blocking is perhaps the most important advantage of proxy servers. Because no TCP/IP packets actually pass between the internal and external networks, a vast number of denial-of-service and exploitation attacks are prevented.

Unfortunately, route blocking is not used often enough. Because many protocols exist for which there are no good proxy services, administrators often must enable routing on the proxy server, which completely eliminates the security gain achieved by route disconnection. If you can, avoid allowing low-level network packets to pass through your proxy server. Most proxy server software will allow you to create generic TCP proxy services for any port using a generic SOCKS proxy or the Unix `redir` utility. These generic proxies, although they cannot perform content filtering, still allow you to keep TCP/IP packets from flowing between your networks.

Logging and Alerting

The final security advantage of proxies is the logging and alerting facilities they provide. Proxies ensure that all content flows through a single point, which gives you a checkpoint for network data. Most proxy software will log the usage characteristics of the proxy by user and can be configured to retain a log of sites they visit. This will allow you to reconstruct the user's web browsing sessions if you suspect some illegal or unethical activity has occurred.

The alerting facility provided by some proxies can alert you to attacks in progress, even though the proxy facility of a server is not generally subject to attack. But the facility can alert you to attempted proxy connections from the external interface, which hackers frequently try to exploit to launder their connections.

Performance Aspects of Proxies

In addition to their security aspects, proxy servers can also perform important performance enhancements:

- Proxies can cache frequently requested data to improve performance by eliminating redundant access to the slower external network.
- Proxies can balance the service load across a number of internal servers.

Caching

As we noted in the beginning of this chapter, proxies were originally developed as a performance improvement, not as a security device. In the early Web days, there were only tens of thousands of sites. They were mostly scientific in nature and didn't change often. Because wide area links to the Internet were slow, a proxy could be used to cache large portions of the Web locally, so internal users could simply browse from the local proxy. Content didn't change very fast, so that made sense.

Caching now only makes sense in those instances where a large number of users frequently access the same web pages over and over. This usage pattern is currently somewhat rare, so the caching aspects of proxy servers are all but obsolete.

As e-commerce becomes more prevalent, caching will again become an important function because many people will perform their jobs by interfacing to a few frequently accessed sites. Consider for example a travel agency that uses Expedia.com and Travelocity.com to perform their work. Many agents would access the same two sites over and over, so caching the main site elements, graphics, and applets would make sense.

Reverse Proxy Load Balancing

Newer proxy servers can be used to "reverse proxy," or provide the proxy service to external clients for internal servers. This functionality is used to balance the load of clients across a number of web servers. Many high functionality websites make use of complex applications in the form of ISAPI applications, Active Server Pages, Java servlets, or CGI applications. These applications execute on the server, so they considerably reduce the number of clients a single server can handle. For example, an NT server running IIS that could reasonably handle 100,000 browsers of standard HTML pages may only be able to handle 5,000 browsers of an ASP page that is executed on the server.

This means that most e-commerce functions cannot actually be handled on a single server, so the site must be run in parallel across a number of machines. For example, `http://www.microsoft.com/` is currently run on 30 identical web servers. DNS provides a rudimentary load-sharing scheme by which subsequent access to a DNS name will provide one of a number of IP addresses, but this does not actually balance the load. Only after a statistically large number of equivalent accesses occurs does this scheme serve to actually balance the client load.

A proxy server can be used to respond to a single IP address and then funnel client connections to one of a number of site servers behind it. The proxy server can use some measurement provided by each web server to maintain awareness of which server has the most remaining capacity. Each connecting client can then be funneled to whichever server has the most capacity to handle it. Because the proxy actually does very little work compared to the effort of serving e-commerce web pages, it can handle many orders of magnitude and more client connections than the e-commerce servers behind it can. Figure 8.2 shows a proxy server load balancing a number of e-commerce servers.

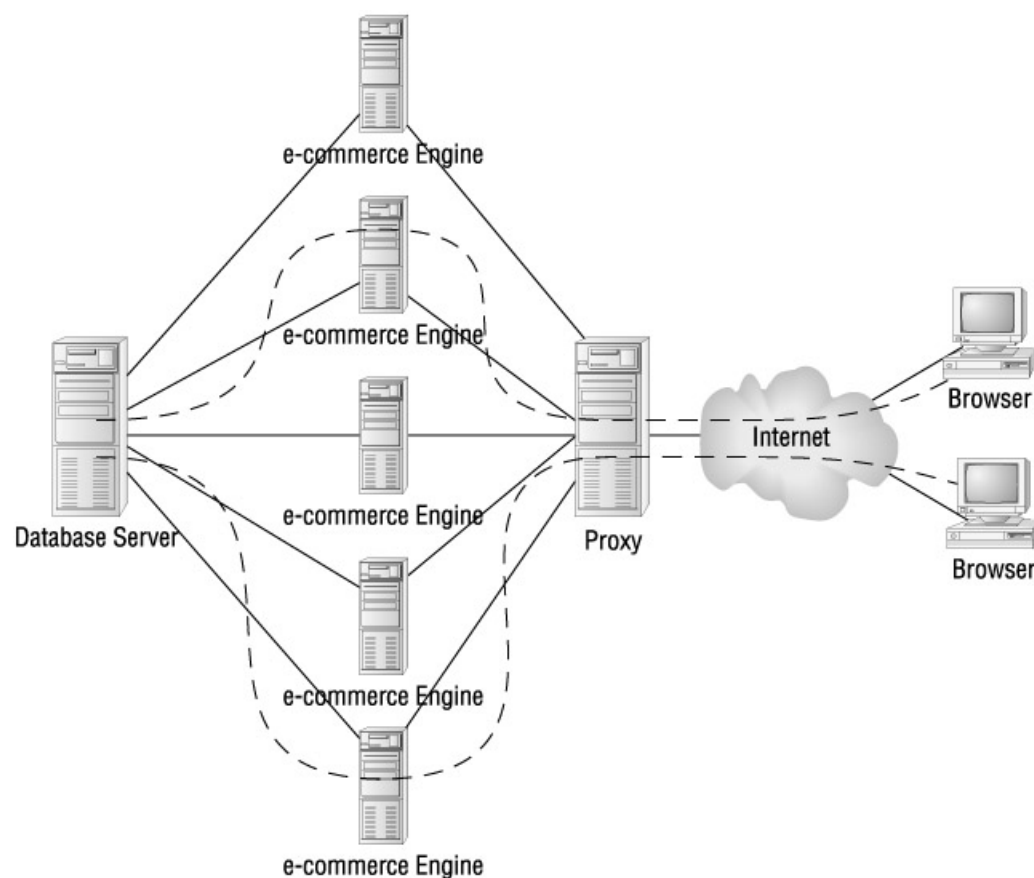


Figure 8.2: Load balancing with a proxy server

Security Liabilities of Proxies

Proxies suffer from some of the following security liabilities:

- Proxies create a single point of failure.
- Client software often must be capable of working with proxies. Only advanced firewall and proxy systems can be configured to work transparently on the network.
- Proxies must exist for each service.
- Proxies do not protect the base operating system.
- Default configurations are often optimized for performance rather than security.

Single Point of Failure

Inherent with any single point of control is a single point of failure. If a hacker can disable your proxy, your entire organization could be cut off from the Internet.

Proxies, routers, and firewalls all suffer from this problem to some degree. With routers the problem is easily fixed by simply having more than one route to the Internet. Firewalls are far more secure than pure proxies because they include low-level packet filtering to eliminate the problems caused by denial-of-service activities. Pure proxy servers do not include the functionality to protect themselves from attack however, so they are very vulnerable both to intrusion and denial of service.

Modern proxy servers usually include a hot-failover feature where a secondary proxy with the same network connections constantly queries the "live" proxy and takes its IP addresses if it appears to have failed. Others use a load-balancing feature to provide multiple peer proxies that are all in use at the same time. The Windows Load Balancing feature of Windows 2000 Advanced Server can be configured with proxy server software to create this type of fault-tolerant proxy.

Clients Must Be Made to Work with Proxies

A proxy-enabled client must exist for each service you wish to proxy. For example, your web browser must support connection to a proxy server by including in the configuration options regarding which proxy service all requests should transmit to. If the client software cannot be configured to use a proxy, a proxy service cannot be used except in conjunction with a true Network Address Translator. This can be a major problem for services like FTP where the client software that ships with most operating systems does not support connection to a proxy server. You can purchase proxy clients for these services, however.

Proxy services included with address translating firewalls can get around this restriction because they can modify inbound and outbound network addresses. This means that clients need not know or be configured to work with proxies that exist as part of a true address translating firewall.

Proxies Must Exist for Each Service

A different proxy service is required for each supported service protocol. Network Address Translation is universal and works with any protocol except those that rely upon payload-embedded IP address information or require the ability to open a back channel to the client. Protocols for which no proxy service is available cannot be connected through a proxy except by a generic TCP proxy service (like the generic SOCKS proxy) that would work much like a Network Address Translator. Any such service would not have the advantage of content filtering, however.

Many services cannot be easily proxied because they require the establishment of a back channel. Only proxy servers that maintain a table of expected return sockets can proxy services like H.323 (the protocol used by NetMeeting for voice and video conferencing).

Many services exist for which there are no effective content filters. Stream-based services like RealAudio or RealVideo are very difficult to filter for content because the content must stream through in real time, and an interruption in the compressed stream will make the remainder of the stream undecipherable. Since content like this cannot be reliably filtered, it should be blocked if considered a security threat.

Proxies Do Not Protect the Base Operating System

Proxy servers are based on web servers, and like web servers, they operate at the Application layer—above the Network and Transport layers. This means that they do nothing to filter TCP/IP packets that arrive at the server, and they don't interfere with other Application layer services like file sharing or remote procedure call interfaces.

This leaves the machine completely open to hacking, unless you take other measures to secure the machine. While most modern operating systems include support for packet filtering, their filters are usually not as robust as true firewalls. And you need to ensure that only those public ports that correspond to the services that you intend to proxy are open.

Some security experts recommend running the fewest possible number of services on a firewall, and separating proxy functionality on to separate machines under the presumption that filters should be as simple as possible to prevent their being exploited. The problem with this is that exploits can occur at any level, and if you put a proxy server behind a filter, the hacker is behind the filter if he exploits the proxy. By using firewalls with integrated proxy servers, the filters can still protect the network even if the proxy service is exploited.

Lax Default Configurations

Many proxy server software packages suffer from lax default configurations that can cause serious security problems. For example, WinGate, the most popular proxy server for home and small office environments, is used to share a single Internet connection rather than for security. For this reason, the software producer made it easy to set up for people who didn't understand proxies, and set it up to work by default for most common protocols.

For versions before 3.0, the default installation opened up a Winsock proxy to the external interface, which allowed hackers to connect to the external interface as if they were internal clients. The hackers could then use the proxy to connect to other web or Internet services as if they were working from the unsuspecting home user's computer directly. This effectively laundered their connection and made it appear as if the owner of the computer running WinGate were performing the illegal activities the hackers actually performed. Version 3.0's default configuration disabled connections coming from the external interface.

Many proxy servers suffer from the problem of lax default configuration because they are often designed for less experienced computer users and put performance and functionality ahead of security. Most can be configured correctly, but users frequently ignore the software once they've got it completely installed.

Performance Liabilities of Proxies

Proxy servers only have one performance liability, but for the sake of uniformity, I'll present it as a bulleted list:

- Proxy servers create a service bottleneck.

Proxies Create a Network Bottleneck

Like firewalls or routers, a single proxy server connection to the Internet can create a bottleneck if it's not properly upgraded as the number of network users' increases. Although proxies initially improve performance through their caching mechanism, you'll make everyone wait behind a slow

machine if you've got more clients than the server can effectively support. But beware of blaming your proxy for a bottleneck that's actually caused by a slow Internet pipe. If you have only one Internet connection, and it is a T1 (1.5MB) or slower connection, any computer that actually meets the minimum requirements for the operating system and the proxy server is fast enough to handle the load. Proxy bottlenecks only occur on network connections faster than 1.5MBs or when something is actually wrong with the proxy server.

This problem is easy to solve—add more proxy servers. Unlike websites or public servers, a proxy server doesn't need to have the exact same configuration across a number of machines. You can directly attach any number of proxy servers to your external network connection and assign each client inside your network to one of the servers on a random or fair share basis. For example, if you've got four proxy servers running, just assign every fourth client to the same proxy server. You'll lose some of the caching effect because a client on a different proxy who accesses a site won't make that site available to the other proxies.

You can also use sophisticated, high availability software and TCP/IP load balancing to handle the connection to multiple proxies, but that involves considerable expense and is not much more efficient. It does provide proxy redundancy though, because otherwise a segment of users would lose service if their assigned proxy went down.

Explicit vs. Transparent Proxies

Most proxies, especially common HTTP proxies, require that the client software be explicitly configured to use the proxy server to access data (such as web pages) from outside the network. This means that not only must every web browser, FTP client, or videophone application you want to proxy have the ability to use a proxy server (many do not, having been naively programmed to expect unfettered access to the Internet), but also a system administrator must either configure all the applications on the client computers in the network to use the proxy or teach the users how to do it.

The configuration issue is such a burden to network administrators that modern web browsers have the ability to automatically detect proxy settings on a network. Other client software, such as FTP or Net2phone, has not been reprogrammed to do this however. But while this feature is a boon to network administrators of web browsers, there is a better way that works for other protocols too and does not require configuring or modifying client network software—transparent proxies.

Transparent Proxies Rewrite the Rules

All modern firewalls can redirect incoming requests to certain ports to specific interior computers that will satisfy those requests (such as a web server on the interior network that is protected by the firewall). Similarly, a firewall can intercept and redirect *outgoing* traffic to a particular computer, such as a proxy server for web requests. The client computer need not know that its traffic has been intercepted because the firewall can redirect the proxy server's response back to the originating client as though nothing untoward had happened (using the same network address translation mechanisms that are now so wide-spread). There are instructions on the Internet for using the firewalling features of BSD or Linux along with separate proxy packages such as Jigsaw in this manner.

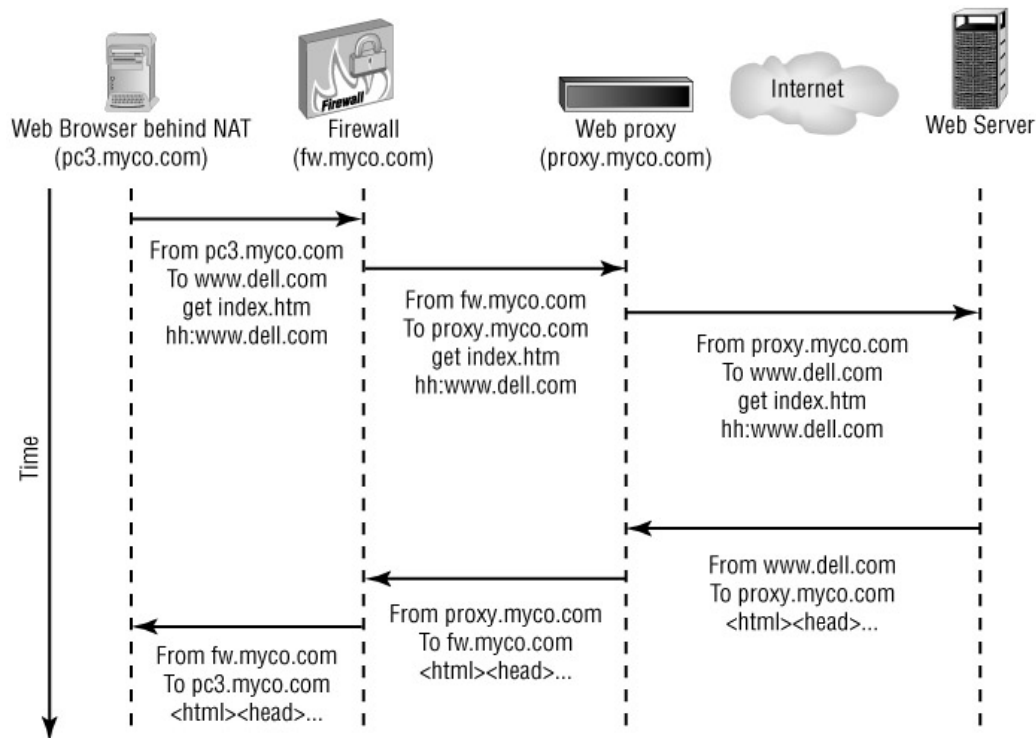


Figure 8.3: Transparent Proxies intercept and forward client requests without requiring client configuration.

Proxy Best Practices

Proxies are useful for a number of different purposes, and for that reason security often takes a backseat to performance or connection multiplexing. Proxies can be extremely dangerous if they're used incorrectly (okay, people can't actually get hurt—just dangerous in the legal risk sense) because hackers can exploit them to make it appear as if their activities are coming from within your network. This can make your company liable for their activities.

This section details proxy servers' best practices that you should implement whenever you use proxy services to connect to the Internet.

Use a Real Firewall

The most important thing you can do to protect yourself is to either use the proxy functionality of a real firewall or put a firewall in front of your proxy server to protect it. There's no reason why a proxy server has to be directly connected to the external network unless the proxy is used for reverse proxy load balancing of a website. Figure 8.4 shows the strong firewall use of proxy servers both for client proxy and reverse proxy functions. The firewall's purpose is to protect the proxy server from exploitation or attack from the network since pure proxies do not include protective features like packet filtering or network address translation.

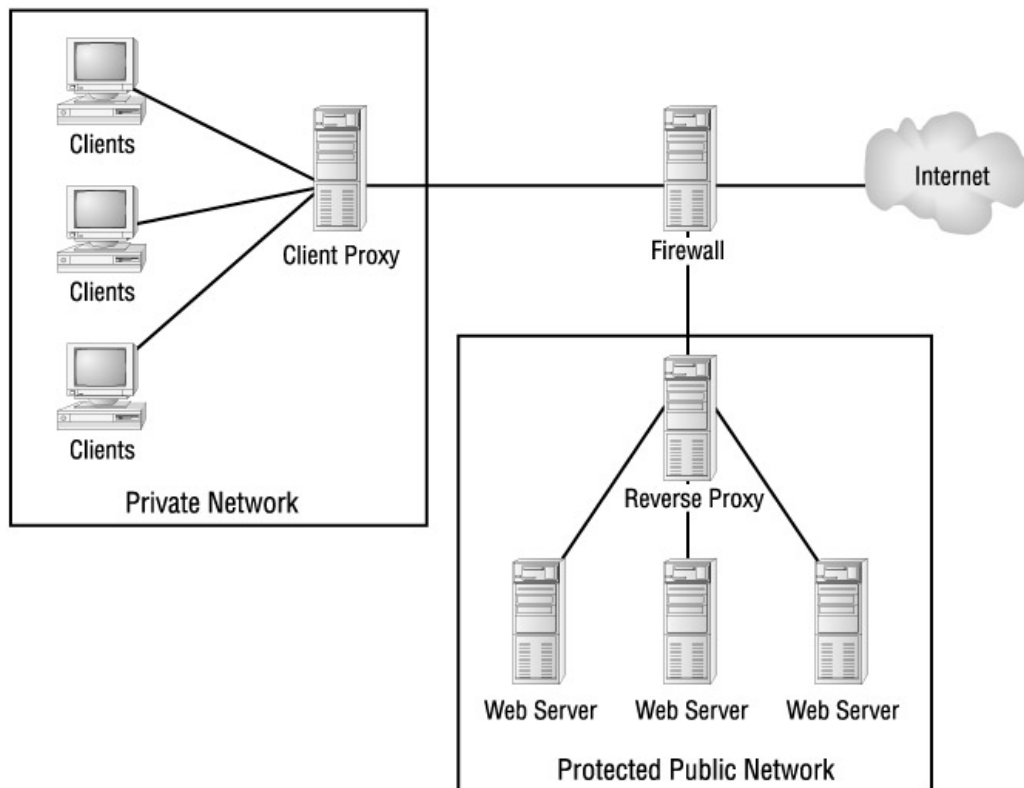


Figure 8.4: Place proxies behind firewalls

Disable Routing

If you use proxies as your primary protection against hackers on the Internet, be sure you disable routing through the proxy. If you allow routing through the proxy, the proxy is not performing a significant security function for your network because your clients will all be directly addressable from the Internet. Proxies' client-hiding feature relies upon disabled routing to prevent a number of low-level protocol attacks.

Proxies are usually set up initially with routing disabled, but after some time a service or protocol might be needed for which you do not have a specific proxy service or which cannot be proxied. Don't be tempted to simply enable routing in this case. If you find you need services that cannot be proxied, use Network Address Translation. If the service can neither be translated nor proxied, don't use it at all.

Secure the Base Operating System

Securing the base operating system is crucial to the effective use of proxies as security devices. If hackers can exploit the server upon which your proxy runs, they can reconfigure the proxy security settings to bypass it completely.

This is especially important in Unix and Windows environments. Both operating systems are notoriously susceptible to well-known hacking exploits, so proxies that run upon them are just as susceptible.

Use strong user-based security permissions as well as port and protocol filtering at the operating system level to make sure your proxy server is serving only those protocols you intend for it to serve. Stay up to date on the latest hacking exploits for your operating system and be certain that you apply patches and hotfixes to your external security servers as they are released. It's more important for a publicly exposed server to be secure than it is to be stable. A crash due to an

untested patch or hotfix only causes a temporary loss of service—it doesn't allow a security breach.

Disable External Access

Never allow external network clients to proxy through your server, even if it seems like it would make sense for remote users to do so. By allowing external proxy access to your server, you make it possible for hackers to exploit your proxy server to launder their IP connections and make it appear as if your proxy server is the origin of their attacks. This could make you legally liable for the damages they cause.

Disable Excess Services

Don't pile all your public services on the same machine as your proxy server. This general rule is especially important when applied to security mechanisms like proxy servers. If a service like FTP or SMTP allows a hacker access to your proxy server, the hacker can disable the proxy server's security settings to gain further access to your network. If these services are divided amongst several machines, however, an FTP-specific attack will only yield access to the FTP server—not the rest of the network.

On Windows NT, it's especially important to unbind the NetBIOS session ports from the external TCP/IP interface through the network Control Panel. Leaving these ports open will make it possible for hackers to use automated password guessing tools to attempt to log directly into your proxy server. Once that's accomplished, they have free rein to modify your security settings.

Case Study: Guilt by Association

A small-business client of mine used a copy of WinGate to share their cable modem among a group of five computers at one of their offices. Cable modems are constantly connected Internet interfaces that operate asynchronously at a fairly high speed—up to 10Mb down from, and 768Kb up to the ISP. They provide more than enough bandwidth to share in a small office.

WinGate is simple enough that they set the software up themselves. Running on a Windows NT Workstation, they established web and e-mail (POP and SMTP) proxies.

After about a year, they noticed that their Internet connection had become comparatively slow, and that the computer seemed to be doing quite a bit of hard disk access. They called me to determine why. I enabled the WinGate logging mechanism and immediately determined that the proxy was working overtime with a number of WinSock connections—a function that they did not use. Within a few seconds, I determined that their proxy server was being exploited by hackers to launder their IP address through an unsecured WinSock proxy service that WinGate established and allowed access to on the external interface by default.

Hackers love WinGate. By running through someone else's proxy server, a hacker can launder their IP address and keep from being found. If they're caught hacking, the IP address shows up in logs as the address of the proxy server, not their own. By scanning for the default WinGate administration port (808) that the default installation leaves open to the outside interface, they can quickly find hosts for their parasitic activities. They scan the address ranges of ISPs that they know provide constantly connected services with static IP addresses (like cable modem providers) so they can exploit the proxy whenever they want.

We recorded all the incoming address information we could, and then I set about the task of properly configuring the WinGate proxy for security by disabling connections from the external

interface and shutting down the WinSock proxy. I warned my client that they may be subjected to numerous denial-of-service attacks because the hackers who'd been routinely exploiting their server would probably be angry that the server had been secured—and that they'd been caught.

Once I had the hacker's IP addresses, I set about contacting their ISPs to get their service shut off. Using tracert and the InterNIC website, I located the actual bottom tier ISPs (those that provide the dial-up service directly to the customer) and contacted them by phone. We'd determined that two of the three hackers were in Canada and one was from Israel. I called the support number for the Israeli company and explained the situation to their support technician. He escalated the call to their network administrator, who told me flat out that they would not pursue the issue even enough to find out which of their customers was to blame, and that I had no legal recourse. Upsetting as that was, he was right. There's no effective way to sue a foreign network service provider to force them to identify their customer in countries where that's not required by law.

Canada is a different story, however. I called their ISP and was greeted by much the same initial response of being elevated to a network administrator. This time the problem wasn't one of indifference as much as incompetence. The administrator admitted that they had no time accounting system or logging in place; they simply billed all their customers a flat rate because they couldn't figure out how to bill for time used or even to keep track of when which users were using what. He admitted that they had no way to determine which customer had been online or using a specific IP address at any specific time. When I warned him that the law made his company responsible for how their network is used, he assured me that they were working on an accounting system that would take care of the problem.

So although I was able to secure the proxy and prevent its future exploitation, I was not able to hold the hackers who'd exploited it liable for their actions.

Chapter 9: Virtual Private Networks

Overview

Using the Internet to link LANs and give remote computers LAN access causes serious security, performance, reliability, and management problems. Your LAN is a protected environment that is only available to members of your organization. The LAN clients and servers should be protected from the Internet by a firewall and proxy servers so that (ideally) network intruders can't even identify their existence, much less target them for individual attack. In order to make it more difficult for hackers to capture private company information, most firewalls are configured not to pass typical LAN service protocols such as NetBIOS, the NetWare Core Protocol, or NFS.

It is possible to directly link your Windows, Unix, or Macintosh based LANs together over the Internet. By simply configuring your firewall to pass NetBIOS, NFS, or AppleTalk traffic, you could allow your employees to have remote access to file and print services. But this would allow hackers to access your data, simply by providing a valid account name and password or by attacking the protocol to exploit a bug that would allow access. Worse, they could read the data you transmit by simply installing a sniffer on any of the Internet networks through which your data flows.

Exposing your LAN file-sharing protocols (like NetBIOS, NFS, or AppleTalk) in this manner effectively makes the whole Internet your LAN. It is not private. Not only could your sales force print to your engineering department's printers or log on to your accounting department's file server, anyone on the Internet could print to the printer or log on to the file server. An intruder would have to guess a password, of course, but hackers have a lot of experience in guessing passwords.

The solution to this problem is Virtual Private Networks (VPNs). VPNs are a cost-effective way to extend your LAN over the Internet to remote networks and remote client computers. VPNs use the Internet to route LAN traffic from one private network to another by encapsulating the LAN traffic inside encrypted IP packets. The encrypted packets are unreadable by intermediary Internet computers and can contain any kind of LAN communications, including file and print access, LAN e-mail, Remote Procedure Calls, and client/server database access.

Virtual Private Networks between LANs can be established using server computers, firewalls, or routers. Client access to the VPNs can be made using VPN software on the client computers or by dialing in to ISPs that support the VPN protocol. Using this second method, however, makes the ISP a partner in your network security in the same way that relying on an ISP for your firewall does.

Pure VPN systems do not protect your network—they merely transport data. You still need a firewall and other Internet security services to keep your network safe. However, most modern VPN systems are combined with firewalls in a single device.

Virtual Private Networking Explained

Virtual Private Networks solve the problem of direct Internet access to servers through a combination of the following fundamental security components:

- IP encapsulation
- Cryptographic authentication
- Data payload encryption

All three components must exist in a true VPN. Although cryptographic authentication and data payload encryption may seem like the same thing at first, they are actually entirely different functions and may exist independently of each other. For example, Secure Socket Layer performs data payload encryption without cryptographic authentication of the remote user, and the standard Windows logon performs cryptographic authentication without performing data payload encryption.

IP Encapsulation

When you plan to connect your separated LANs over the Internet, you need to find a way to protect the data traffic that travels between those LANs. Ideally, the computers in each LAN should be unaware that there is anything special about communicating with the computers in the other LANs. Computers outside your virtual network should not be able to snoop on the traffic exchanged between the LANs or be able to insert their own data into the communications stream. Essentially, you need a private and protected tunnel through the public Internet.

An IP packet can contain any kind of information: program files, spreadsheet data, audio streams, or even other IP packets. When an IP packet contains another IP packet, it is called IP encapsulation, IP over IP, or IP/IP.

Why encapsulate IP within IP? Because doing so makes it possible to refer to a host within another network when a direct network connection may not exist. IP encapsulation can make it appear to network computers that two distant networks are actually adjacent—separated from each other by a single router. But they are actually separated by many Internet routers and gateways that may not even use the same address space because both internal networks are using address translation.

IP encapsulation also makes it possible to refer to a remote host within the 10 and 192.168 reserved blocks that are not routable over the Internet. When the 10.0.0.0/8 and the 192.168.0.0/16 reserved network blocks were assigned, routing rules were created to ensure that they could not be routed over the Internet backbone in order to provide a modicum of security and to prevent conflicts with other networks using the same address block; nobody would be able to reach hosts in that network range over the Internet, so if you used those ranges in your private network you would be reasonably safe from hacking and could guarantee that your address space didn't conflict with anyone's public addresses. These rules are still in effect today and work pretty well for their purpose, but if you want to connect remote offices and use the reserved address blocks, you have to encapsulate them within IP packets that can be routed on the public Internet.

The tunnel end point—be it a router, VPN appliance, or a server running a tunneling protocol—will receive the public IP packet, remove the internal packet contained within it, decrypt it (assuming that it's encrypted—it doesn't have to be), and then apply its routing rules to send the embedded packet on its way in the internal network.

As an example, consider two IP networks linked by a router. Both are /24 (class C) IP subnets, one with the network address of 10.1.1 and the other with the network address of 10.1.2. In this example, the fourth number in each network is reserved for the station address, in the range 1 to 254. The router must have a network interface adapter on each network so that it can move IP traffic between the two LANs. The .1 and .254 station addresses are typical addresses reserved for routers and gateways, so in this network the router has one adapter with the IP address of 10.1.1.1 and another with the IP address of 10.1.2.1. All of the computers in both networks have a netmask of 255.255.255.0, since both networks are /24 subnets.

When a computer in the 10.1.1 network (for example, 10.1.1.23) needs to send an IP packet to a computer in the 10.1.2 network (such as 10.1.2.99), the communication proceeds as follows:

1. The originating computer first notices that the network portion of the destination address (10.1.2.99) does not match its own network address.
2. Instead of attempting to send the packet directly to the destination, the originating computer sends the packet to the default gateway address for its subnet (10.1.1.1).
3. The router at that address reads the packet.
4. The router determines that the packet should be placed on the 10.1.2 network subnet.
5. The router sends the packet from its adapter (10.1.2.1) to the destination address (10.1.2.99) on that network subnet.
6. The destination computer reads the packet.

In comparison with the preceding example, consider two IP networks linked by routers using IPSec. One LAN has the network address 10.1.1.0, and the other has the network address 10.1.2.0. In this example, the IPSec routers on each network provide the network connection to the Internet. One IPSec router has a LAN IP address of 10.1.1.1 (assigned by the IT staff) and an Internet address of 250.121.13.12 (assigned by the ISP), while the other has a LAN IP address of 10.1.2.1 and an Internet address of 110.121.112.34.

Communication in the IPSec-connected LANs starts and ends the same way it does in router-connected LANs. The IP packets have further to go, though, so more work is done in the middle. Compare the following example to the earlier one:

1. The originating computer (10.1.1.23) first notices that the destination address (10.1.2.99) is not in the same subnet as itself.
2. Instead of attempting to send the packet directly to the destination, the originating computer sends the packet to the default gateway address for its subnet (10.1.1.1).
3. The IPSec router on the 10.1.1.0 network reads the packet.
4. The IPSec router on the 10.1.1.0 network determines that the packet should be placed on the 10.1.2.0 network subnet, for which it has a security association defined that points to the 110.121.112.34 router over the Internet.
5. The IPSec router sends an IKE (Internet Key Exchange) negotiation request to the remote router, authenticates its identity, and negotiates for a set of encryption and authentication keys.
6. The IPSec router encrypts the packet and encapsulates it in another IP packet.
7. The router sends the encapsulated packet from its WAN interface, which is connected to the Internet (250.121.13.12), to the Internet address (110.121.112.34) of the IPSec router of the 10.1.2 network subnet.
8. The IPSec router of the 10.1.2 network subnet reads the encapsulated and encrypted packet from its Internet interface.
9. The IPSec router of the 10.1.2 network decrypts the embedded IP packet. The fact that its keys can decrypt the packet verifies that it is a valid packet that has not been tampered with and that it comes from a trusted source that held the encryption keys.
10. The IPSec router of the 10.1.2 network sends the packet from its adapter (10.1.2.1) to the destination address (10.1.2.99) on that network subnet.
11. The destination computer reads the packet.

Note that from the point of view of the two network client computers, it doesn't matter how the packet got from one IP subnet to the other. As far as the network client computers are concerned, a single router is the same thing as two IPSec routers and a VPN connection.

Cryptographic Authentication

Cryptographic authentication is used to securely validate the identity of the remote user so the system can determine what level of security is appropriate for that user. VPNs use cryptographic authentication to determine whether or not the user can participate in the encrypted tunnel, and may also use the authentication to exchange the secret or public key used for payload encryption.

Many different forms of cryptographic authentication exist:

- **Secret Key Encryption** Also called Shared Secret encryption, relies upon a secret value known to both parties. Simply knowing the value proves to the provider that the requester is to be trusted. Challenge and response can be used to make sure that only hashes of the secret, not the secret itself, are transmitted on the network, and one-time password variations can be used to ensure that the secret changes each time it's used.

The problem with secret key encryption comes from key exchange; there's no truly secure way for both parties to simultaneously "know" the secret key without some sort of non-encrypted initial exchange. Usually, that exchange takes place in the mind of a single individual who sets the same secret on two different end systems.

The algorithms necessary to perform secret key encryption are magnitudes (at least three of them) faster than the algorithms necessary to perform public key encryption. In most production systems, public key encryption is used to securely authenticate and set up a communication channel, and that channel is then used to exchange a set of secret keys securely, so that a high-speed secret key encryption algorithm can be used.

- **Public Key Encryption** Relies on the exchange of unidirectional keys—keys that can only be used to encrypt data or to decrypt data, but not both. This means that the decryption key (the private key) is held on the receiver and never transmitted over a public network, which makes the encrypted data secure during its transmission because it can't be decrypted by anyone else, even if they have the encryption key (the public key). Tunnel-end systems may exchange pairs of public keys to form a bidirectional channel, or the public key receiver may encrypt a shared secret key and transmit it to the public key transmitter to use for future communications (because secret key encryption is faster than public key encryption).

If a hacker intercepted the public or encrypting key, the only thing they could do with it is encrypt data and transmit it to the receiver; they would not be able to decrypt the contents of data they intercept.

Data Payload Encryption

Data payload encryption is used to obfuscate the contents of the encapsulated data without relying on encapsulating an entire packet within another packet. In that manner, Data Payload Encryption is exactly like normal IP networking, except that the data payload has been encrypted. Payload encryption obfuscates the data, but does not keep header information private, so details of the internal network can be ascertained by analyzing the header information.

Data payload encryption can be accomplished using any one of a number of secure cryptographic methods, which differ based on the VPN solution you chose.

Characteristics of VPNs

When you consider establishing a VPN for your company, you should understand the advantages and disadvantages of VPNs when compared with traditional LANs and WANs, namely:

- VPNs are cheaper than WANs.
- VPNs are easier to establish than WANs.
- VPNs are slower than WANs.
- VPNs are less reliable than WANs.
- VPNs are less secure than isolated LANs or WANs.

These characteristics are detailed in the following sections.

VPNs Are Cheaper Than WANs

The primary consideration, in many cases, is that a VPN is often much cheaper than a WAN of similar size, especially when you want Internet connectivity for your LANs. A single dedicated leased line between two major cities costs many thousands of dollars per month, depending on the amount of bandwidth you need and how far the circuit must travel. A company's dedicated connection to an ISP is usually made with a leased line of this sort, but the circuit is much shorter—usually only a few miles—and an IP connection is usually already in place and budgeted for. With a VPN, only one leased line to an ISP is required, and it can be used for both Internet and VPN traffic. ISPs can be selected for proximity to your operation to reduce cost.

You can also establish a business Internet connection to some ISPs using regular analog modems, ISDN, xDSL, or cable modems, depending on the services available in your area. These methods of connecting to the Internet can be much less expensive than dedicated leased telephone lines, but you must evaluate whether the bandwidth provided by these connection methods is sufficient for use with your VPN. Satellite connections are usually provided as hybrid systems that are frequently incompatible with higher-end cryptographic protocols like IPsec, so be sure you know the details of your connection methodology before you choose it.

VPNs really shine compared to traditional WANs in the way they deal with LANs being connected over widely separated geographical areas (in different cities, different states, or even different countries). The costs of dedicated long-distance leased lines are much greater than the costs for local-loop circuits (connections between locations that share the same local telephone switch). Because you can use the Internet instead of expensive long-distance lines to create a VPN, you can eliminate the long-distance circuit expense.

When considering a VPN, you have to take into account your total monthly bandwidth requirements as well as your peak, short-period bandwidth requirements. Many ISPs apply a surcharge when the total amount of data transferred in a month exceeds a certain amount. It is unlikely that even intensive long-term use of the Internet connection will exceed the costs of leasing a long-distance line of similar capacity, but you should still check the policies of the ISPs in your area and make plans accordingly.

The VPN advantage in terms of remote user dial-in is that you don't have to provide and support your own specialized dial-in equipment, such as modems and terminal servers, or maintain dial-up phone lines. You can rely on an ISP to provide that service for you. The management and equipment depreciation costs alone should justify paying for your users' ISP accounts (and many users have arranged for their own ISP accounts for their home computers, anyway).

Reality Check: Frame Relay vs. VPN

My company recently migrated a client's network from an International Frame Relay WAN to a VPN. The company used Frame Relay circuits with a 256Kbps Committed Information Rate (CIR) to connect their offices in Mexico, three sites in England, and the United States. The monthly recurring costs for this 256Kbps circuit amounted to just over \$15,000 per month.

At each location, we installed a T1 rate IP connection from the same global ISP (UU-NET). We then acquired and installed dedicated IPsec VPN routers at each of the premises and configured a "mesh" pattern VPN (where a separate VPN is established between each two VPN routers). We brought the VPN online and tested traffic through it for a week before changing the default gateway address on each internal LAN to point to its VPN router rather than its Frame Relay router. We left the Frame Relay network in place for two months while we validated the reliability of the VPN, and then shut the Frame Relay down.

The total cost of all the IP circuits came to about \$5,000 per month, a third of the monthly cost of the Frame Relay. More importantly, the speed of the WAN jumped from 256Kbps over Frame Relay to 1536Kbps over the IP connections (less about 20% for protocol overhead that the Frame Relay connection didn't incur). Reliability is slightly lower with the VPN due to the fact that the management software reloads all the VPN routers whenever a remote user account is added to the VPN, but this is easily managed by scheduling updates during network off-peak periods or by switching from a mesh to a less efficient hub-spoke network where all VPN devices have a single security association with a central VPN router.

VPNs Are Easier to Establish

The two most difficult issues in WAN creation and management have to do with establishing communications links over the dedicated leased phone lines (using specialized communications devices) and routing WAN traffic over those links using specialized routers. It typically takes at least two months to get a traditional WAN established using dedicated leased lines or Frame Relay.

When you establish a VPN over the Internet, your Internet Service Provider will help you make the initial IP connection to their service. Once you configure your firewall for tunneling, you can let the Internet do your routing for you. You don't have to learn how to program and manage specialized routers and gateways (unless you use them in your local network, as when several LANs in a campus are linked together). You do have to establish and manage the VPN connections, however, and you must maintain a connection to the Internet (you will most likely be maintaining an Internet connection anyway).

In any case, when an existing IP connection exists, a VPN connection can be established in mere hours.

Reality Check: Business Velocity

My company recently took on a new client that had just created their business by purchasing three business units from other companies. Two units were in the U.S., St. Louis and San Diego, and one was in Guadalajara, Mexico. The new company was given 90 days to be off the original owners' networks and have their own system implemented.

Reengineering and integrating three completely separate LANs in separate cities is not trivial, especially when you are under an extremely constraining deadline. We used VPN technology to get the WAN portion setup quickly and easily.

By utilizing each unit's existing IP connection, we were able to establish a running and workable VPN within the first few days. We then ordered new IP circuits that would be delivered over the course of the next eight weeks. Because the businesses already had IP connections that would be running for the 90-day interim, we were able to create the WAN immediately and begin reengineering the LANs without waiting for physical circuit installation, which is always slow.

When the circuits arrived one at a time at each of the facilities, we simply moved the VPN devices over to them and reestablished the same VPN connection. To the LAN users, the connections simply got faster and more efficient as our own circuits were brought online. We had completed the network migration well in advance of the deadline, which passed unnoticed.

VPNs Are Slower Than LANs

You will not get the same performance out of your VPN that you would with computers that share the same LAN. Typical LANs transfer data at 10 or 100Mbps while the Internet limits VPNs to the slowest of the links that connect the source computer to the destination computer. If, for example, the dial-in PC is connected to the Internet by a 56Kbps modem, then data transfer will go no faster than 56Kbps. If your LANs are connected to the Internet through the ISP with T1 leased lines, then the maximum you can expect for inter-LAN traffic is 1.5 Mbps (each way). Of course, WANs are no different; if you linked the same LANs directly via T1 leased lines, you would still have a 1.5 Mbps (each way) bandwidth limit.

Even if you have a very fast connection to your ISP, you may not get to use the full bandwidth of that connection because there may be a lower-speed link between your ISP and the ISP that serves the remote LAN. You may connect to the ISP via FDDI (a 100Mbps medium), for example, but the ISP may only have a T3 connection to the Service Access Point that links ISPs, limiting your total throughput to about 45Mbps. Also, you must share the Internet bandwidth with other Internet users, which means you actually get considerably less than the maximum theoretical bandwidth.

Furthermore, you may find that Internet congestion between your VPN end points will put a serious drag on your network. If your VPN travels over more than 20 intermediate systems between end points, as shown by the `tracert` command, it will probably be uselessly slow, especially if those 20 systems are owned by different major ISPs. The best way to take care of this problem is to use the same national or global ISP to connect your systems. This way, all your data will travel over their private network, thus avoiding the congested commercial Internet exchange network access points.

Reality Check: The Importance of Using the Same ISP

I have a client who recently established VPN service between their U.S. facilities and their facilities in Europe. The client had an existing circuit with AT&T CERFnet and needed service for the new European facility.

Rather than simply accept the ISP with the lowest bid price, I cautioned my client to check the network timing between their existing facility and the various candidate ISPs in Europe. We had each candidate provide the IP address for a router of theirs in the target city and performed ping and traceroute tests against the IP address throughout the day.

Not surprisingly, the AT&T affiliate in Europe had traceroute timings under 45ms, while the next best candidate averaged well above 130ms. The difference between these times is dramatic and can be the difference between success and failure in establishing a reliable VPN. My client went with the AT&T affiliate despite the slightly higher cost and now has a very reliable VPN.

I use the following timing charts to determine how well a VPN circuit is likely to perform in the real world:

- < 40ms—Perfect
- < 80ms—Good
- < 120ms—Workable
- < 160ms—Poor
- > 160ms—Useless

Remember to use a tool like FreePing or NeoTrace with larger ping packets (1450 bytes is a good MTU size) to perform averaging throughout the workday, to make sure you're not just seeing a particularly good or bad pattern and to let the various intermediate systems get their caches established similarly to the way a VPN would work.

VPNs Are Less Reliable Than WANs

With a WAN, you retain far greater control over your network than you do with a VPN. When you use a WAN, you configure the routers and gateways, you negotiate the leased line service for the entire distance between your LANs, and you configure and maintain the specialized devices that maintain the WAN. You do not share the WAN bandwidth with any other organizations or individuals. With a VPN, on the other hand, you delegate all of those decisions to individuals outside your company. You and the ISP cooperate to establish the first stage of the VPN (from your LAN to the ISP), but someone else manages every other stage. This means that you have less control in the event of network outages. Also, unexpected surges in Internet activity can reduce the bandwidth available to users of your VPN. On the other hand, these service providers often have considerable experience in solving communications problems and can fix them much more quickly than the typical LAN administrator. Also on the plus side, Internet capacity has been growing consistently.

The reliability problem is most effectively dealt with in the same manner as the speed problem: Use the same national ISP throughout your network. If you have remote home users, make sure your ISP also provides commercial dial-up service so those users won't have to route through the CIX NAPs to get to your network. By using a single national ISP, you only have one company to hold accountable when things go wrong—you won't have competing companies pointing the finger at each other in an attempt to avoid responsibility.

Reality Check: 99% Reliability

In my experience, VPNs are about 99% reliable—meaning you should expect 3.65 days of downtime per year. The reason has to do mostly with reconfiguration and with some vagaries of the IPsec protocol, fragile implementations of IKE, and the inherent reliability problems with the Internet.

I have a client whose VPN took a while to achieve reliable operation. They used an outsourced VPN service that managed and updated their VPN devices, which were not integrated with their firewalls. The service established a mesh pattern VPN despite the fact that 90% of all traffic came and went

through their single headquarters facility, making their use an excellent candidate for the much more reliable hubspoke architecture. Due to a few misconfigurations, miscommunications, and the unreliability of this service's update methodology, each time a change was made to repair functionality or bring a new remote office online, something else would invariably break, leading to a few harried days of reconfigurations. Their boxes also seemed to need an inordinate number of firmware updates because they would freeze and (according to the service provider) need to be erased and reflashed.

Add to this the fact that each and every remote user addition required every device in the network to be reconfigured, and you can see the situation they were in.

VPN devices, protocols, and platforms are not all created equally—nor are they necessarily compatible just because they claim to use the same set of encryption protocols. Before you deploy a specific VPN system, be certain you've tested a pair of evaluation units thoroughly and force the vendor to put you in touch with a customer who has widely deployed and used the boxes. If they refuse or can't seem to put you in touch with a large client, it means that they either don't have any or they don't want you talking to their existing clients—both reasons to avoid their product, in my opinion.

VPNs Are Less Secure Than Isolated LANs or WANs

One drawback of VPNs (and the point that is most relevant to this book) is that they are less secure than LANs and WANs that are not connected to the Internet. Before a hacker can attack your network, there must be a way for the hacker to reach it. But how many LANs or WANs today are not connected to the Internet? A VPN is marginally more vulnerable to network intrusion than a LAN or WAN that is connected to the Internet because the VPN protocol is one more interface for the hacker to try to subvert.

VPN solutions like Microsoft's implementation of PPTP provide attack vectors for hackers to exploit. Any method that provides access to your network is a potential vector for attack, and VPNs force you to expose a vector into your network on the public Internet. Strong encryption and authentication measures can nearly eliminate the risk of intrusion, but they can't completely eliminate it. And any flaws in the implementation of a VPN protocol will considerably weaken your security posture.

Remote access clients present special security problems. Although they connect securely to your computers, they also participate directly on the Internet. This makes the remote client vulnerable to attack from the Internet, since client operating systems are far less robust than server operating systems in most cases. Because very few strong security packages exist to secure individual clients, you will have difficulty ensuring that a remotely connected client doesn't present a hole in your security posture. Firewall vendors have just identified this potential vulnerability, and a few of them offer strong firewall services for some types of client computers. Your best bet to prevent this problem is to allow only Windows NT /2000/XP, Mac OS X, or Unix remote hosts that have been secured as strongly as their native security will allow to attach to your network. Traditional Macintosh and Windows 95/98/Me clients are not capable of providing the level of security required for facilities that need a strong security posture.

Reality Check: A Tunneling Worm

During a recent network integration session, I had the opportunity to work side-by-side with the integration team for a large ERP system vendor. They used specially secured Windows NT Workstation laptops to connect securely through their IPSec VPN to the company intranet. Their public servers had strong firewalling and used eEye's hardened security filter for IIS to prevent Internet worms from exploiting their servers. Their intranet servers needed no such security because they were not attached to the Internet.

Or were they? A few people in the company had taken it upon themselves to install Windows 2000 on their own laptops—because it's better for laptops, of course. And which installed IIS as a standard component, which was not secured, which contracted the Code Red virus while attached to the Internet and the VPN, which traveled through the VPN to the intranet servers and infected them, which subsequently infected every IIS machine in the company from the inside.

Worm is an apt name for these exploits. They will worm their way through any possible chink in your security armor. I've warned clients for years that unsecured VPN clients are a theoretical backdoor into an otherwise secure network, and with the advent of virulent Internet worms, the theoretical part of that statement is gone. Your security perimeter needs to be extended to every computer that has access to your VPN, and you must be able to control the software configuration on those machines. Warn your users now that unsupported modifications to their laptops are a serious security threat that cannot be tolerated.

Types of VPNs

The example of an IPSec routed VPN connection described earlier in this chapter is simplistic because it doesn't include such common Internet security components as firewalls and proxy servers. It does represent one type of VPN, however: the router-based VPN. Many firewalls also include IP tunneling functionality on which a VPN can be based. Many medium-to-large networks use routers to manage traffic routed within, as well as into and out of, the LAN. A number of VPN-only routers also exist; these devices are called VPN appliances because they perform only one function. Regardless of how the VPN is set up, any properly secured network with an Internet connection will include firewall and possibly proxy server services. A VPN must be configured to work with these services.

There are three types of VPNs:

- Server-based VPNs
- Firewall-based VPNs
- Router-based VPNs including VPN appliances

Server-Based VPNs

In a Windows-based network, perhaps the easiest and least disruptive way to establish a VPN between LANs is to dedicate a Windows server computer to routing the VPN traffic. Existing firewall, router, and proxy server services can be left in place, and the only modification to the Internet security setup required is for the firewall to pass the encryption ports through to the Windows RRAS (Routing and Remote Access Service) server. Windows NT supported only PPTP, but Windows 2000 includes support for PPTP, L2TP, and IPSec.

Linux and the open-source community have brought strong security tools like Blowfish, Free S/WAN, and PPP over SSL to the Unix world, as well as support for PPTP and L2TP. Many Unix

vendors are incorporating IPSec into their own TCP/IP implementations as well. These products can all be used to create VPNs between LANs using general-purpose computers.

While Windows 2000 is a full-featured operating system that can run firewall software, maintain encrypted links using numerous protocols, and provide file and print services to network clients all at the same time, it is not a good idea to do all that with just one computer. A security failure in any one of these services would compromise the entire network instead of just the affected computer. The RRAS server, for example, doesn't have to be a privileged computer in the network; all it has to do is encapsulate and un-encapsulate network traffic. A hacker who has compromised a properly isolated RRAS server will still have regular LAN security to defeat (such as usernames and passwords) and will have to get through the firewall to get to the RRAS server. If the file server also hosted the RRAS services, the hacker would have access to all of the network files immediately.

You should be aware that VPN traffic destined for a remote network in the VPN travels over each LAN twice—once in the form of regular LAN traffic to the RAS server, and once again in encapsulated form from the RRAS server to the firewall. While the duplication of LAN traffic is inefficient, the amount of traffic is usually insignificant compared to regular LAN traffic because the bandwidth of the Internet connection limits the amount of information that can be sent over the encrypted link.

Windows 2000 Server comes with everything you need to establish a Virtual Private Network over the Internet using PPTP, L2TP, or IPSec, but their implementation of PPTP is flawed and should only be considered somewhat secure. Use IPSec for the best possible security if you are going to use Windows 2000 to create a VPN. You may want to rely on additional software to protect your network from IP intrusion, even if the RRAS service that comes with the operating system is sufficient to establish an encrypted link between secure LANs.

Linux's IP masquerade/IPChains features can be used with additional open-source software to create fairly robust VPNs as well. However, integrating numerous packages from different vendors isn't easy to do correctly, and can lead to a "Swiss-cheese" security effect, where all the pieces are in place but holes exist because they aren't well integrated.

Firewall-Based VPNs

Every LAN that is connected to the Internet needs a firewall to isolate LAN traffic (NetBIOS traffic in the case of NT networks; NFS, Telnet, or X-Windows in the case of Unix networks; AppleTalk in the case of Macintosh networks; and NCP in legacy NetWare networks) from Internet traffic. A firewall should block all ports that aren't expressly required to provide service—especially the NetBIOS, NFS, Telnet, or X-Windows ports—from being accessed from outside your network, and should specify which computers inside your network are allowed to access the Internet.

That is not all that modern firewalls can do, however. Popular firewalls can perform address translation; take care of protocol and port filtering; redirect common services such as mail, news, and FTP; and even proxy such protocols as HTTP, SMTP, NNTP, Telnet, and FTP. Since firewalls already do every other sort of analysis and transformation of network packets, it is a simple matter to include IP tunneling capability in the firewall.

Originally, the tunneling protocols included with most firewalls were proprietary and would only establish a VPN link with the same brand of firewall on the remote LAN or with client software written specifically for that firewall. This situation is changing rapidly with the widespread adoption of the IPSec+IKE (IP Security with Internet Key Exchange) encryption and negotiation protocols. Although many vendors now support IPSec+IKE encryption, their specific versions are not always

compatible. If you intend to use IPSec+IKE in a multi-vendor firewall network, contact each vendor to make sure they've tested their software to work with your other firewalls and to identify any configuration issues you'll have. A completely standardized implementation of IPSec+IKE should eliminate the compatibility problems caused by proprietary encryption systems.

Many of the firewalls include VPN software for individual remote client computers to connect to the firewall and establish a tunnel. If you need to connect remote computers to your LAN, you should check to make sure the client software is available for all your supported platforms. You also need to run computer-based firewalling software to secure the otherwise unprotected computer from being exploited while it's attached to your VPN.

Router-Based VPNs

Large networks (such as those in a business, school, government, or campus environment) often comprise several LAN segments linked together by routers. The routers isolate internal LAN segment traffic while conveying inter-LAN traffic quickly and efficiently. The routers are custom hardware devices with specialized circuitry and programming for handling network packets.

Simple routers merely transport the network packets from one segment to another, but the more complex and expensive routers can also act as firewalls, examining the network traffic and manipulating it (blocking ports, redirecting packets, and so on) according to rules established by the network administrator. Some routers even include the ability to encapsulate network traffic and establish VPN links between routers. IBM's 2210 router family, Cisco's routers running IOS, and Ascend's MAX switches are three popular router solutions that support VPN capabilities.

VPN Architectures

Given a number of participants in a VPN, there are three basic methodologies you can use to create your VPN:

- Mesh, where each participant has a direct security association with every other participant
- Hub and spoke (also called a star), where each participant has a single security association with a central VPN router that has a security association with every VPN device
- Hybrid VPN, which is a combination of the two forming either a mesh of hubs or a star of hubs

Of the two, mesh is more efficient and hub and spoke is more reliable.

Mesh VPNs

Mesh VPNs work by creating a specific connection between every VPN participant. This requires an SA (security association or VPN definition) for every VPN device on your network to be present in every device.

When a VPN device needs to transmit data, it consults its routing/SA table to determine which VPN connection should be used to transmit the data, which is then sent over the Internet directly to the remote VPN end point.

Creating a large mesh VPN can be problematic because each SA requires memory on every device, which means that the least of the devices in your network must have enough flash memory

for the entire set of devices in your VPN. For most users, this isn't much of an issue because modern devices have enough capacity to handle numerous connections.

The larger problem with mesh VPNs is the requirement that every VPN device on the WAN must be updated every time a new VPN end point is added, which can be quite often in a growing network. If you treat remote users' computers as full participants in the VPN, this can be quite problematic indeed.

Remote VPN updates are problematic in themselves because the update data must be transmitted through the existing VPN, and then the device must go offline, perform the update, and come back up on the VPN. If something goes wrong with the update or if there's a flaw in the new security update, the device may not come back up on the VPN, which will then require local reconfiguration. Unless you have someone at every branch office capable of bringing the device up manually, this is a serious problem. And let's face it—if you had that, you probably wouldn't be attempting the update remotely.

Mesh architectures are most suited to stable businesses that are not experiencing rapid growth and where traffic flow between sites is unpredictable.

Hub and Spoke VPNs

Hub and spoke VPNs are the simple solution to the problems created by the frequent update problems caused by mesh VPNs.

Hub and spoke VPNs concentrate the complexity of the network onto a single central VPN router. Each remote device has only one security association to the central device, and new devices and remote users need only to be added to that central device. Once a remote device has established its connection to the central device, no further configuration will be necessary.

Unfortunately, the cost is that every packet must flow into and out of a central VPN device, which, depending upon the geographic separation of your network, could send packets many milliseconds out of their way when compared to the direct link provided by a mesh. Furthermore, the central VPN router will incur double the bandwidth cost for each connection as it flows in and then out the same pipe, and it must be prepared to talk to every connection at its full speed. This means that your central router should have as much bandwidth as the sum of all other VPN participants doubled—a number that can get large and expensive very fast.

Furthermore, your central VPN router must be able to route between security associations internally. Surprisingly, many VPN devices don't have the ability to do this, so be sure to check your vendor's documentation before you purchase if you think you'll need to use a hub and spoke architecture.

Hub and spoke architectures are suited best to small and medium-sized businesses that have a central office and a number of branch offices, because in this environment the majority of the traffic flows to and from the central office anyway.

Hybrid VPNs

Hybrid VPNs are the pragmatic combination of hub and spoke VPNs and mesh VPNs. Most VPN implementations actually wind up being hybrids, whether by design or by accident as they grow. There's nothing wrong with this, and there's no reason to stay dogmatically with one architecture or the other. Each should be used where the advantages provided by its strengths are outweighed by

the disadvantages presented by its weaknesses.

In general, the stable "core" VPN of your business should be implemented as a mesh because of its decreased load characteristics. New additions to the business, rapid expansions, and remote users should be handled at the edges of the mesh by hub and spoke extensions, so that their addition and deletion doesn't impact the entire mesh. New branches can then be added to the mesh at a semi-annual VPN update session where you've planned for additional support and scheduled the possibility of downtime that may be associated with the update. Generally, you should always add new offices as spokes to the closest hub, and then schedule a general update to integrate all the spokes at regular intervals.

In my opinion, remote users should always be handled as spokes from a hub, and, unless there's significant bandwidth reasons not to, from a single VPN router irrespective of where they connect from geographically. Remote user connectivity can then be centrally controlled, managed, and if necessary, cut off at a single point in your VPN.

Common VPN Implementations

Although theoretically any cryptographically strong algorithm can be used with some form of IP encapsulation to create a VPN, a few market-leading implementations have arisen, popular either because they are easy to hack together from separate existing tools, because they are the agreed upon standards of numerous small vendors, or because a large vendor implemented them and incorporated them for free into ubiquitous products like operating systems.

The common VPN implementations are:

- IPSec Tunnel Mode
- PPTP
- L2TP
- PPP/SSH or PPP/SSL

Each of these common implementations is detailed in the following sections.

IPSec

IPSec is the IETFs standards suite for secure IP communications that relies on encryption to ensure the authenticity and privacy of IP communications. IPSec provides mechanisms that can be used to do the following:

- Authenticate individual IP packets and guarantee that they are unmodified.
- Encrypt the payload of individual IP packets between two end systems.
- Encapsulate a TCP or UDP socket between two end system(hosts) inside an encrypted IP link (tunnel) established between intermediate systems (routers) to provide virtual private networking.

IPSec performs these three functions using two independent mechanisms: Authenticated Headers (AH) to provide authenticity and Encapsulating Security Payload (ESP) to encrypt the data portion of an IP packet. These two mechanisms may be used together or independently.

Authenticated Headers works by computing a checksum of all of the TCP/IP header information and encrypting that checksum with the public key of the receiver. The receiver then decrypts the

checksum using its secret key and then checks the header against the decrypted checksum. If the computed checksum is different than the header checksum, it means that either the decryption failed because the key was wrong or the header was modified in transit. In either case, the packet is dropped.

Note Because NAT changes header information, IPSec Authenticated Headers cannot be reliably passed through a network address translator (although some NATs can perform translation automatically for a single internal host). ESP can still be used to encrypt the payload, but support for ESP without AH varies among implementations of IPSec.

With Encapsulating Security Payload, the transmitter encrypts the payload of an IP packet using the public key of the receiver. The receiver then decrypts the payload upon receipt and acts accordingly.

IPSec can operate in one of two modes: *transport mode*, which works exactly like regular IP except that the headers are authenticated (AH) and the contents are encrypted (ESP), or *tunnel mode*, where complete IP packets are encapsulated inside AH or ESP packets to provide a secure tunnel. Transport mode is used for providing secure or authenticated communication over public IP ranges between any Internet-connected hosts for any purpose, while tunnel mode is used to create secure links between routers or other network end points for the purpose of linking two private networks.

Microsoft recommends IPSec for use in gateway-to-gateway communications, and does not consider IPSec suitable for client-to-gateway communications, because the protocol itself does not include user authentication. For that reason, Microsoft's implementation of IPSec does not include support for IPSec tunnel mode. Their recommendation is usually astute in this case.

The root of the problem is that client and machine authentication are handled separately; use of the correct machine gains access to the network, and then a separate authentication protocol gains access to private network resources. A hacker could exploit the window between machine authentication and user authentication to run probes against the network without logging in or to run automated password attacks against the network.

Consider the following example: A legitimate network user uses an IPSec-configured laptop to gain access to the local network, and then simply logs in to the domain over the connection. While traveling, the laptop is stolen at the airport and winds up in the hands of an opportunistic hacker. The hacker uses commonly available password patching tools to gain local access to the laptop, and discovers that it is automatically connecting to a remote network via IPSec. This provides the hacker remote access to the private network, so he uses network-scanning tools to "sniff out" the structure of the network. He then finds a client computer inside the remote network. Keep in mind that while he has IPSec access to the remote network, he has still not authenticated with it or logged in.

So he runs a network-based password-guessing program over the course of a few days to guess the password of the user who lost the laptop.

When user authentication is combined with machine authentication at the security gateway, this attack is not possible because access to the private network is not granted unless both the machine and the user are authenticated.

The bias against IPSec as a client-to-gateway technology is somewhat esoteric, because it can encapsulate secure authentication methods like Kerberos to provide user authentication, or it could incorporate support for client-side authentication by requiring a user identity to decrypt the

machine's IPSec identity. When an IPSec tunnel is established between a client system and an IPSec gateway, and then a secure authentication protocol like Kerberos is used to secure authentication of the user at the client station, security remains intact. While a gap between machine authentication and user authentication exists which could technically be exploited, this level of security is sufficient for most business purposes.

We've used IPSec for client-to-gateway authentication since shortly after it became available in this mode, and found it to be both more reliable and more secure than PPTP. Microsoft recommends using IPSec over L2TP for client-to-gateway authentication.

Tunnel Mode

In a normal routed connection, a host transmits an IP packet to its default gateway, which forwards the packet until it reaches the default gateway of the receiver, which then transmits it to the end host. All the computers in the connection have to be in the same public address space.

In IP over IP or IP/IP, the default gateway (or another router down the line) receives the packet, notices that its route for that packet specifies an IP/IP tunnel, so it establishes a TCP/IP connection to the remote gateway. Using that connection, the gateway transmits all of the originating hosts' IP traffic inside that connection rather than forwarding it. IP over IP (or IP/IP) is useful for virtual networking, so that private IP addresses (in the 192.168.0.0 range, for example) can be passed over the public Internet.

Many IPSec implementations allow both IP/IP and IPSec/IP. IP/IP provides a non-encrypted virtual tunnel between two end systems (which can use AH to guarantee authenticity), and IPSec/IP uses ESP to encrypt the payload of the carrier IP, thus encrypting the entire encapsulated IP packet. Why would you use a non-encrypted tunnel between end systems? Because you aren't passing particularly sensitive information and your end points are compute-limited. But these days with processor speeds far in excess of what standard users require, there's little reason to use clear channel tunnels.

Internet Key Exchange (IKE)

IPSec uses public key cryptography to encrypt data between end systems. In order to establish an IPSec connection to a receiving host, the transmitting host must know that host's public key. Technically, the transmitter could simply ask the host for a public key, but that doesn't provide authentication—any host could ask for the key and get it. This is how SSL works; the identity of the machine is not important, and SSL relies upon some other protocol to authenticate the user once the tunnel is established. That's fine for public websites, but IPSec was designed from the outset to perform host authentication through knowledge of appropriate keys.

Note IPSec is defined by RFCs 2401 through 2409, along with numerous other RFCs that describe supporting protocols.

IPSec uses the concept of the Security Associations (SA) to create named combinations of keys and policy use to protect information for a specific function. The policy may indicate a specific user, host IP address, or network address to be authenticated or to specify the route for information to take.

In early IPSec systems, public keys for each SA were manually installed via file transfer or by actually typing them in. For each SA, each machine's public key had to be installed on the reciprocal machine. As the number of SAs a host required increased, the burden of manually keying machines

became seriously problematic. IPSec was used primarily only for point-to-point systems because of this.

IKE obviates the necessity to manually key systems. IKE uses private key security to validate authority to create an IPSec connection and to securely exchange public keys. IKE is also capable of negotiating a compatible set of encryption protocols with the foreign host, so that administrators don't have to know exactly which encryption protocols are supported on the opposite host. Once the public keys are exchanged and the encryption protocols are negotiated, an SA is automatically created on both hosts, and normal IPSec communications can be established. With IKE, each computer that needs to communicate via IPSec needs only to be keyed with a single private key. That key can be used to create an IPSec connection to any other IPSec host that has the same private key.

Tip In Windows 2000, IPSec policies can be configured to use Kerberos to automatically exchange private keys for IKE. This eliminates the need for manual keying, and allows for completely automatic secure encryption between members of the same Active Directory in Windows 2000 networks.

The IKE initiator begins an IKE request by sending a plaintext connection request to the remote host. The remote host generates a random number, keeps a copy, and sends a copy back to the initiator. The initiator encrypts its private key using the random number and sends it to the remote host. The remote host decrypts the private key using its kept random number, and compares the private key to its private key (or list of keys, called a *key-ring*). If the private key does not match, the remote host will drop the connection. If it does match, the remote host will encrypt its public key using the private key and transmit it back to the initiator. The initiator then uses the public key to establish an IPSec session with the remote host.

Like nearly everything in Windows 2000, IPSec configurations are managed by creating a policy. For IPSec, you create a different policy for each receiving computer that has unique IPSec requirements. Destination networks, computers, and tunnel end points are specified by IP address. For each IPSec policy, you can define the exact encryption protocols used by the policy and the keying method used for IKE.

Layer 2 Tunneling Protocol (L2TP)

Layer 2 Tunneling Protocol (L2TP) is an extension to the Point-to-Point Protocol (PPP) that allows the separation of the data link end point and the network access point. In traditional PPP, a user (typically a dial-up user) establishes a PPP connection with a remote access server. That server answers the Data Link layer connection (modem call) and also serves as a network access point by removing the data encapsulated in the PPP message and forwarding it on the destination network. The encapsulated data might be an Apple-Talk frame, IP packet, IPX Packet, NetBIOS packet, or any other network layer packet.

Note In this section, I use the generic term "remote access server" to refer to any L2TP compatible remote access server, which could be a Unix computer, a Windows 2000 computer, or a dedicated remote access appliance. In Windows 2000, this service is called RRAS.

L2TP separates the call answering from the network access routing. With L2TP, a caller can call a modem bank (or DLS Access Module or whatever), and that equipment can simply encapsulate the received L2TP packets into Frame Relay, ATM, or TCP/IP packets for further transport to a remote access server. Upon reaching the remote access server, the L2TP packets are unwrapped and their payload is forwarded onto the local network.

L2TP is designed to allow for less expensive ISP equipment by separating the remote access server functions into a hardware function (physically receiving the connection) and the software function (gating the encapsulated PPP data) that can be performed on separate machines. This has a number of important benefits:

- Users can dial a local modem bank that will forward L2TP to a distant RAS server, thus avoiding long-distance charges for direct dial remote access.
- L2TP payloads can be encrypted using IPSec to provide secure authenticated remote access.
- Multilink L2TP sessions can be physically answered by different receivers and correctly bonded on a single RAS server. With current PPP multilink, all the channels must be connected to the same RAS server.

L2TP can use IPSec to encrypt PPP frames, thus providing what amounts to a secure PPP session for remote users. L2TP was specifically designed to provide remote user authentication and connection to remote networks. Because L2TP is an extension of PPP, any network layer protocol (such as IPX, NetBEUI, or AppleTalk) can be embedded inside L2TP. In contrast, PPTP and IPSec are both specific to IP networks and do not function with other protocols. The use of PPP also provides support for all of the standard user authentication protocols, including CHAP, MS-CHAP, and EAP.

L2TP uses UDP rather than TCP/IP as its data transport because the embedded PPP protocol can provide the necessary guarantee of reliability for the stream. L2TP operates over UDP port 1701.

Microsoft considers L2TP to be the ideal user authentication mechanism for completing the strong security requirements of remote access users. Because L2TP provides secure user authentication, and then uses IPSec to provide machine authentication, it provides the ability for remote users to connect without the authentication gap that would occur if user authentication were separated from secure tunnel establishment.

PPTP

PPTP was Microsoft's first attempt at secure remote access for network users. Essentially, PPTP creates an encrypted PPP session between two TCP/IP hosts. Unlike L2TP, PPTP operates only over TCP/IP, whereas L2TP can operate over any packet transport, including Frame Relay and ATM. PPTP does not use IPSec to encrypt packets; rather it uses a hash of the user's Windows NT password to create a private key between the client and the remote server that (in the 128-bit encrypted version) is salted with a random number to increase the encryption strength.

L2TP is the successor to PPTP; it is more generalized in that it works over any packet transport, and its encryption strength is far stronger thanks to IPSec encryption. PPTP should be used for legacy compatibility, but new installations should favor L2TP for secure remote access.

Open-source developers for Unix implementations including Linux and the various free BSD derivatives have implemented PPTP to support creating inexpensive encrypted tunnels with Windows clients. Both client-side and server-side implementations are available that interoperate well with Microsoft's implementation of PPTP. So while IPSec is still the future of VPNs, PPTP is a pragmatic "here now" solution to cross-platform VPN interoperability.

PPP/SSL or PPP/SSH

PPP (Point to Point Protocol) over SSL (Secure Socket Layer) or SSH (Secure Shell) are two common methods that Unix and open-source operating system administrators employ to create VPNs "on the fly." Both methods, which might be considered "hacks" in the Windows world, employ a clever combination of an existing encrypted transport (SSL or SSH) and an existing tunnel provider (PPP).

Point to Point Protocol (PPP)

PPP was originally designed to support multi-protocol transport over serial lines. Originally, the dial-up access world was clearly split into specific operating system camps: Windows, which supported only NetBIOS connections over modem links; Macintosh, which supported only AppleTalk connections; Unix, which supported only Serial Line IP (SLIP) connections; and NetWare, which supported only IPX connections to NetWare servers. PPP was developed originally to abstract the protocol away from the connection so that a serial line connection could be established that would then be able to carry any network layer protocol. So, essentially, PPP creates a Data Link layer-connection between end points over which a Network layer-protocol can be transported—or, in other words, it creates a tunnel.

Because of its flexibility, PPP can be used to create a connection between any two IP systems, and then transport IP over the PPP connection. This is an easy way to create IP/IP tunnels without specific operating system support for tunneling. But PPP performs no encryption; so while tunneling is useful, it's not secure.

Secure Socket Layer (SSL)

Secure Socket Layer (SSL) is a public key encryption protocol developed by Netscape to support secure web browsing. SSL does not perform authentication; its only purpose is to encrypt the contents of a connection between a client and a public server. So SSL performs an essentially "pure" public key exchange—when a client connects to the SSL port on a server, the server transmits an encryption key, which the client uses to encrypt its data stream. The client does the same thing, so a bidirectional secure stream can be established. This stream is used to exchange a pair of randomly generated secret keys so that high-speed encryption algorithms can be used.

Secure Shell (SSH)

SSH is the Unix secure shell, which was originally designed to shore up the serious security flaws in Telnet. Telnet allowed users to connect to a Unix host and establish a remote text console from which the host could be operated. Because Telnet hails from those early days when hackers did not have access to the Internet, it performs no encryption and only simple unencrypted password challenges. SSH shores this up by performing secure authenticated logons using perfect forward secrecy and then by encrypting the communication session between the client and the host. Like most Unix applications, SSH can accept redirection to and from other running applications by correctly constructing "pipes" on the Unix command prompt. Unlike SSL, SSH uses secret key encryption, so both parties must know the secret key in advance to establish a connection.

Securing PPP

Given the PPP command built into most modern implementations of Unix and either SSH or SSL, it's a simple task to construct a command that can direct the establishment of an encrypted tunnel and pipe its input and output streams to the PPP command. This, in essence, creates a virtual

network adapter on each host system that is connected via PPP to the remote host, which is in turn encrypted by either SSH or SSL.

The security of a system like this is based mostly on the security of the underlying cryptosystem—SSL or SSH. If the administrator has done his homework and knows for certain the identity of the hosts involved in the connection, these connection methods are as secure as PPTP or L2TP.

Note Although the implementation differs in unimportant ways, PPTP is essentially PPP over SSL, and provides basically equivalent security.

Secure Remote Access

Virtual Private Networks are great for connecting LANs, but what about people with isolated computers such as telecommuters, roving troubleshooters, salespeople, executives on the move, or anyone else lucky (or unlucky) enough not to work in an office or cubicle?

The traditional (read: expensive) way to provide these users with LAN access is to install modem banks and purchase phone lines so that they can dial up to your LAN using modems. Dial-up services provided in this manner require a modem and a phone line for each simultaneous dial-in connection supported. If you want two people to be able to connect at the same time, you'll need two modems and two phone lines for your dial-up server. If you want to support two hundred people at the same time, you'll need two hundred modems and two hundred phone lines (and some esoteric serial connection hardware as well). Also, either your company or the dial-up users will have to pay any long-distance toll charges if the users aren't a local phone call away from the dial-up server.

Just about anywhere you can go in the industrialized world today, an Internet Service Provider is a local phone call away. Internet service from these providers is relatively cheap because the ISPs can spread the cost of supporting dial-up connections across a wide base of customers. It makes sense to use these dial-up services just for your own network connections rather than to duplicate them. The problem is this: How do you protect the network communications between the remote computer and the computers on your LAN?

There are two ways to extend a Virtual Private Network to include individual remote computers connecting over the Internet. One way is to have your users dial up to ISPs that include a VPN port in their dial-up service, essentially making the ISP a partner in your LAN security management. The other way is to move the VPN port into the remote computer.

VPN in the ISP

Internet Service Providers use special devices called remote access switches, remote access servers, terminal servers, or serial concentrators to connect a large number of phone lines and modems to their dial-up network. The serial concentrators allow a server computer (often a dedicated VPN router or Unix host but sometimes a Windows server computer) to accept a large number of dial-in connections. The remote access switches, servers, and terminal servers are special-purpose computers designed just to connect dial-in users to the network. In either case, the dial-up server (a general-purpose computer or specialized device) performs the functions of authenticating the user and connecting the user's computer to the ISP's LAN.

For remote access, Cisco, Microsoft, and many other vendors are throwing their weight behind a new protocol called L2TP. When the user establishes an account with the ISP, the user (in cooperation with the network administrator of the LAN the user wants to connect to) can specify which VPNs the user's computer should be allowed to connect to. When the user connects to the remote access switch (typically using the PPP protocol), the ISP's remote access switch first gets the user's ISP account name and password, and then makes an encrypted connection over the Internet to the RRAS server specified by the user's network administrator. The remote user can (after providing a valid account name and password for the LAN, of course) then participate in the LAN like any other network client.

Having the ISP establish the encrypted tunnel connection as well as authenticate and encapsulate the remote users' network traffic provides equivalent security to having your LAN's ISP manage your firewall for you. Many companies rely on the ISP to provide firewall services, and this can be a cost-effective solution when you have confidence in the security and responsibility of the Internet service provider. However, as mentioned elsewhere in this book, depending on people outside your network to keep it secure is better than nothing but not a good idea if you're serious about security.

VPN in the Dial-Up Client

In most cases, the remote user will not be able to rely on the Internet service provider to establish a VPN session with your remote users. The vast majority of firewall and encrypted tunnel vendors provide small client-side versions of their tunnel software that can be run directly on the remote access client. This allows the client to connect directly to the firewall over the Internet and appear as if it is a workstation on the local network.

In order for the client computer to establish a VPN session, it must first connect to the Internet. This connection can be made through any ISP or can even be made from a computer on a foreign LAN (one that's not a part of your VPN) that is connected to the Internet. Once the client computer is on the Internet, the client VPN software can establish the encrypted connection to your firewall or server using TCP/IP.

Watch out for the hidden problems of network translation in an IPSec connection. IPSec cannot be network address translated (okay, a single IPSec connection that uses only ESP tunnel mode can, but not more than one). This causes serious problems for some traveling users because hotels that provide Internet service frequently deploy network address translation to expand their IP range. Satellite providers and other alternative access providers deploy split upload/download schemes that interfere with the proper operation of IPSec clients as well.

If you run into these problems, you might want to consider using PPTP. Although it's leaky, Microsoft has fixed most of its most egregious problems, and, unlike all the other remote access methods, it translates well through NAT and works behind just about any IP connection irrespective of what kind of trickery is being used to actually get to the Internet. This is about the only reason I would ever recommend using PPTP.

VPN Best Practices

Virtual Private Networks are convenient, but they can also create gaping security holes in your network. The following practices will help you avoid trouble.

- Use a real firewall.
- Secure the base operating system.

- Use a single ISP.
- Use packet filtering to reject unknown hosts.
- Use public-key encryption and secure authentication.
- Compress before you encrypt.
- Secure remote hosts.

Use a Real Firewall

As with every other security component, the best way to ensure that you have comprehensive security is to combine security functions on a single machine. Firewalls make ideal VPN end points because they can route translated packets between private systems. If your VPN solution weren't combined with your NAT solution, you'd have to open some route through your firewall for the VPN software or the NAT software, both of which could create a vector for attack.

Real firewalls are also most likely to use provably secure encryption and authentication methods, and their vendors are more likely to have implemented the protocol correctly. Ideally, you'd be able to find an open-source firewall whose source code you (and everyone else) could inspect for discernable problems. The remainder of this book discusses various firewall solutions to help you decide what will work best in your environment.

Secure the Base Operating System

No VPN solution provides effective security if the operating system of the machine is not secure. Presumably, the firewall will protect the base operating system from attack, which is another reason why you should combine your VPN solution with your firewall.

Implementing any sort of VPN end point on a server without also implementing strong filtering is asking for trouble—without a secure base operating system, the VPN can be easily hacked to gain access to your network from anywhere.

Use a Single ISP

Using a single ISP to connect all the hosts acting as tunnel end points will increase both the speed and security of your tunnel, because ISPs will keep as much traffic as they possibly can on their own networks. This means that your traffic is less exposed to the Internet as a whole and that the routes your ISP uses will avoid congestion points in the Internet. When you use multiple ISPs, they will most likely connect through the commercial Internet exchange network access points—the most congested spots on the Internet. This practically guarantees that your VPN tunnel will be slow—often uselessly slow for some protocols.

Choose an ISP that can also provide dial-up service to your remote users who need it. Alternatively, you may choose a local ISP that is downstream from your national ISP because they are also on the national ISP's network and many national ISPs don't provide dial-up service.

Use Packet Filtering to Reject Unknown Hosts

You should always use packet filtering to reject connection attempts from every computer except those you've specifically set up to connect to your network remotely. If you are creating a simple network-to-network VPN, this is easy—simply cross-filter on the foreign server's IP address and you'll be highly secure. If you're providing VPN access to remote users whose IP addresses change dynamically, you'll have to filter on the network address of the ISP's dial-up TCP/IP domain. Although this method is less secure, it's still considerably more secure than allowing the entire

Internet to attempt to authenticate with your firewall.

Use Public–Key Encryption and Secure Authentication

Public key authentication is considerably more secure than the simple, shared secret authentication used in some VPN implementations—especially those that use your network account name and password to create your secret key the way PPTP does. Select VPN solutions that use strong public–key encryption to perform authentication and to exchange the secret keys used for bulk stream encryption.

Microsoft's implementation of PPTP is an example of a very insecure authentication method. PPTP relies upon the Windows account name and password to generate the authentication hash. This means that anyone with access to a valid name or password (like a malicious website one of your users has visited that may have initiated a surreptitious password exchange with Internet Explorer) can authenticate with your PPTP server.

Compress Before You Encrypt

You can get more data through your connection by stream compressing the data before you put it through your VPN. Compression works by removing redundancy. Since encryption salts your data with non–redundant random data, properly encrypted data cannot be compressed. This means that if you want to use compression, you must compress before you encrypt. Any VPN solution that includes compression will automatically take care of that function for you.

Secure Remote Hosts

Make sure the remote access users who connect to your VPN using VPN client software are properly secured. Hacking Windows home computers from the Internet is depressingly easy, and can become a vector directly into your network if that home computer is running a VPN tunnel to it. Consider the case of a home user with more than one computer using a proxy product like WinGate to share his Internet connection, who also has a VPN tunnel established over the Internet to your network. Any hacker on the planet could then proxy through the WinGate server directly into your private network. This configuration is far more common than it should be.

The new breed of Internet worms like Code Red, Nimda, and their derivatives are running rampant on the cable modem and DSL networks of home users right now, where they find a garden of unpatched default installations of IIS, Microsoft's notoriously insecure web server. These clients are suddenly the Typhoid Marys of the corporate world, propagating worms to the interior of corporate networks through their VPN connections.

Alert users to the risks of running proxy and web service (or any other unnecessary service) software on their home machines. Purchase client firewalling software to protect each of your home users; remember that a weakness in their home computer security is a weakness in your network security when they're attached to your network.

Prefer Compatible IPSec+IKE VPNs

To achieve the maximum flexibility in firewalls and remote access software, choose IPSec+IKE VPN solutions that have been tested to work correctly with each other. IPSec+IKE is the closest thing to a standard encryption protocol there is, and although compatibility problems abound among various implementations, it is better than being locked into a proprietary encryption protocol that in turn locks you into a specific firewall vendor.

Keep in mind that roaming users will probably not be able to get their software IPSec clients running from behind a network address translator, and that these are becoming very common in hotels, Internet cafés, and other places that provide broadband service for roaming users. If you've got this problem, the only easy solution is PPTP.

Case Study: VPNyone?

A client called me in to evaluate why his users were having such a hard time exchanging mail with their main office. Their San Diego facility consisted of a small, leased office space for ten users with a single server and a firewall connected to a 56K ISDN connection. They used PPTP to connect over the Internet to their main office in Cleveland. They ran MS Exchange Server at both sites and had them configured to exchange e-mail through the PPTP tunnel. The server in San Diego was configured as a Backup Domain Controller to the PDC in Cleveland. Other traffic transmitted through the tunnel included Windows NT domain control synchronization, the occasional remotely opened file, and license control software information for the application packages that they used. Central license managers in Cleveland were relied upon to allow access to locally installed applications.

They complained of excessive e-mail transmission times and in some cases, altogether missing e-mail messages.

Checking the event logs showed numerous RPC communication failures from the Exchange service, occurring about every 30 minutes. This corresponded to the timeout delay for the Exchange servers to connect to one another to synchronize mail. So, the Exchange servers were failing to authenticate with one another and therefore were not exchanging mail. Early in the morning and late in the day, the servers would correctly authenticate and begin transmitting mail.

I surveyed their usage patterns to get a feel for how much bandwidth they might be using. As it turned out, the entire company was well used to e-mail. The average employee expected to receive upwards of 100 mail messages per day, which they would usually respond to, creating a mail overhead of 200 messages per employee per day. Each message took up an average of 2K bytes including header information, for a daily transmission of about 400Kbps per user per day. Multiplied by 10 users, that figure came to 4MB of mail traffic per 8-hour work period. The only other significant traffic transmitted between the two networks was license information each time certain applications were started, but that data was negligible and it generally worked.

I first began throughput testing through the ISP connection to a fast server in the local area to make sure that the Network layer to the ISP was working fine. It performed flawlessly.

Ping testing the server in Cleveland was disappointing: only about 33% of ICMP packets made the round-trip. Applying that efficiency to the pipe meant that at most, they were getting 33% of the speed of their 56K link, or about 18Kbps.

After talking to the Cleveland office and verifying that their own Internet links could connect at 100% to local hosts in their area, we had determined that it was, in fact, the low connection efficiency of the VPN that was causing the exchange servers to frequently fail when communicating with one another. And, considering the fact that the Exchange servers would transfer no mail until they correctly authenticated, and that there were 30-minute delays between authentication attempts, most of the day would go by without any mail transfers occurring at all.

I then issued a tracert (the Windows version of traceroute) command to check the route between the Cleveland and San Diego servers. The connection between servers traveled through more than

20 intermediate systems from 15 different ISPs and through 3 of the 4 major CIX NAPs. I'd never seen a more circuitous route. It became quite clear that through that many crowded exchanges, a VPN would never get enough data through to be effective.

I then recommended that they switch to a single ISP, so they decided to switch the San Diego office to the same ISP as they used in Cleveland. They had to change from 56K ISDN to 56K Frame Relay, but the circuit cost was only slightly more and the data rate was unchanged. Once the cutover was performed, everything began humming along smoothly. A quick ping check showed 100% of ping packets returning, and a tracert showed only six routers between their servers—and none going through the CIX NAPs.

Chapter 10: The Ideal Firewall

Overview

The chapters at the end of this book review a number of commonly used (and a few less commonly used) firewall packages and toolkits. We'll provide detailed information on the strengths and weaknesses of each, but as firewall implementation changes with time (they get better, usually) only those firewalls that are currently available will be covered. What should you look for when you are examining firewalls? How would you set up the ideal firewall for your network? These are the questions that will be answered here.

First, you'll determine the security needs for your organization and network. Second, you'll see how the ideal firewall should be configured for varying degrees of paranoia. Third, you'll learn about the various ongoing tasks you'll need to perform with even the most automated and secure firewall. Finally, you'll find out what you should do when your network is actually under attack.

This chapter is broken down into two major parts. The first part, "Defining Your Security Requirements," will help you figure out what general type of security your business requires. The second part will then explain exactly how to configure your border gateways to achieve that level of security. You may find that you'll read back and forth between the two sections to gain a full understanding of the problem.

Defining Your Security Requirements

No two networks have exactly the same security requirements. A bank, for example, is going to be a bit more concerned than a retail clothing store about network intrusions. The type of security concern varies as well as the degree—in a University computing lab the administrator is just as concerned about *hosting* the source of hacking attacks as well as being the target of them.

To decide just how much effort to expend in securing your network, you need to know the value of the data in your network, the publicity or visibility of your organization, and the harm that could be caused by loss of service. You should also consider how much disruption or imposition you can live with on your network in the name of security.

Similar organizations have similar requirements, so you can compare the needs of your network to those organizational types listed below.

Home Offices

A home office is the simplest Internet connected network. Usually, a home office has two to three computers connected in a peer-to-peer fashion on a small LAN. These networks either have a modem attached to each computer so users can connect to the Internet or they have one computer or device that mediates access to the Internet whenever any of the users need an Internet connection. Sometimes the device that connects to the Internet is an inexpensive network hub and NAT router.

The typical home office budget can't afford to dedicate a computer to be a network fire-wall. Sometimes the Internet service provider is relied upon to keep the hackers out. However, this is not a particularly effective technique because ISPs vary in competence and workload, and they never customize security to fit your needs—they provide only a "one-size-fits-all" solution that is

necessarily lax because they don't know how their customers will use the Internet.

Just because most firewalls are prohibitively expensive for home use doesn't mean you are helpless. Chapter 13, "Security Utilities" details a number of "mini-firewalls" that are intended to be installed on personal computer directly, as well as popular low-cost NAT routers which provide very strong default security. Small firewall-less networks can still (and should) install current operating system patches to protect the computers from TCP/IP attacks such as Ping of Death and the Out of Band attack. File sharing should be turned off for computers that are connected to the Internet (or, for more advanced operating systems such as Windows NT and Unix, those services should be disconnected from the network adapter or modem that is connected to the Internet). Any unnecessary services should also be turned off so network intruders can't exploit them.

A recent welcome development is the proliferation of devices that include simple stateful packet filters with Network Address Translation and Internet connectivity via dial-up modem or Ethernet connectivity to a cable or DSL modem. These inexpensive devices greatly increase the security of home office networks by hiding the identity of computers on the LAN and by foiling packet-based exploits, but they do not provide the full range of protection provided by full-spectrum firewalls.

The reason home office networks aren't exploited more often is because their network connections are usually intermittent, their connection speed is low so it takes along time to hack into them, and they seldom provide services (such as websites) that hackers can exploit (with the notable exception of home offices that hook Windows computers directly to the Internet and do not turn off file and printer sharing, or Windows NT/2000 computers that leave IIS activated.) Most hackers exploit random targets of opportunity, so a computer that spends most of its time detached from the Internet isn't going to make a very juicy target. The biggest threat to the home office network is from someone who knows about the network and has a specific reason to attack it. Disgruntled or former employees, business competitors, or an individual with a personal axe to grind are the most likely culprits.

Cable modem and DSL users have become a favored target of hackers however, because their connection speeds are high, their connections are always on, and because they often have no security in place, and their computers are left in the default installation state without security patches applied.

The best way to permanently connect a home office telecommuter to a corporate network is to use a small firewall built to do exactly that, like the SonicWALL SOHO. These firewalls are complete, real firewalls that include IPSec and can be remotely managed through the VPN by the corporate IT staff. In this configuration, the home office is just like any other branch office—connected through a VPN to a firewall with a single public IP address and configured to perform Network Address Translation so the connection can be shared by a few computers. Unfortunately, these devices run about \$500n about \$500 each, so they're not particularly cost effective for many users.

The next best way is to use a small NAT device that can pass a single IPSec connection, like the Linksys Cable/DSL Router with IPSec passthrough. In this configuration, the device doesn't come with IPSec, but it will allow a single computer with an IPSec client to establish connections and route through it. It provides the inherent firewalling provided by all NAT devices, and can be used to share a single Internet connection amongst multiple users. Neither your ISP nor the corporate IT group will see anything other than the single IP address of the NAT device, from which all connections including the IPSec connection will appear to come. This configuration is not really remotely manageable by the IT staff without potentially creating security problems, so it's most effective for users with some technical skill. This method will also work with proprietary VPN solutions like PPTP, L2TP, etc. as long as the NAT device can properly translate the protocol. This

solution would cost about \$150 including the price of the hardware NAT device and the license for the IPSec software client.

Small Service Businesses

Small service business networks, with a typical computer count of around a dozen or so, often have a dedicated computer for file and print services and, in many cases, a dedicated connection to the Internet. Although few small service businesses actually have firewalls, they all should. The potential loss of data and business productivity due to a network intrusion more than justifies the cost of one extra computer and some software.

You don't want to go overboard with security in a small service business, however, and very few small service businesses will go to great lengths to bulletproof their networks because a cost/benefit analysis will usually show that less stringent security is sufficient. Consider, for example, a heating and air-conditioning company that has a small network with an Internet connection. The company's computers have little that would interest either a random hacker or a rival company that might engage in industrial espionage. The network users want as few restrictions as possible on how they access the Internet, so it is difficult to justify draconian network policies.

Tip The small service business network administrator should be concerned about security, but the appropriate policy for the firewall is to permit by default, and to specifically deny packets, protocols, and services on the firewall that the administrator judges to be dangerous.

Professional Firms

Like the small service business, a small confidential practice such as a law firm, accounting firm, psychiatry practice, or medical specialist may have a half dozen to a dozen or more computers connected in a LAN with an intermittent or permanent Internet connection. The small confidential practice should have a more stringent security requirement than the typical small business, however, because the practice's computers contain confidential information that invite specific and targeted attack from network intruders over the Internet.

Tip Because of the sensitivity of the information and the attraction this type of network presents to hackers, the network administrator of a small confidential practice should be cautious (denying packets, protocols, and ports by default unless the rules established specifically allow them) or strict (not routing IP packets at all and allowing only proxied network traffic through the firewall) about security.

Manufacturers

A large network with 50 to 100 computers is a much more tempting target to the average hacker, especially if the network has expensive network equipment and VPN links to other large computer networks. This is the type of network used by medium-to-large corporations, and the very size and complexity of corporate networks make them easier for hackers to attack.

Large corporate networks also may be subject to specific targeted attacks for the purposes of industrial espionage or anticompetitive denial of service. Since corporations have more employees (and former employees) than smaller businesses do, the corporations are also much more likely to come under attack from insiders or former insiders.

A corporation with a lot of public visibility (such as Sony, Microsoft, Pepsi, or Disney) also has the problem of hackers trying to penetrate their networks for the greater bragging rights than would be

achieved by hacking other, less well-known companies (such as McMaster-Carr or Solar Turbines).

Tip Network administrators of large corporate networks need to take extra care that their networks are not compromised because the potential cost of lost productivity is proportionately greater in the larger networks than it is in small ones, and because the large corporate network makes a much more tempting target for hackers. A cautious (deny routing by default) or strict (no routing at all) policy is most appropriate for these kinds of networks.

Government Bureaus

The networks used by governmental bureaus have all of the characteristics of corporate networks (they are often large, have interesting hardware, and provide links to other networks), but governmental networks are also tempting targets because of their political nature. The Bureau of Reclamation has little to worry about, but the FBI, on the other hand, is under almost constant siege from the very hackers they chase. As a general rule, the more visible the organization, the more likely it is to attract the ire of a hacker with an agenda.

Tip Network administrators of governmental bureaus should be either strict (allowing no routing) or paranoid (minimizing any sort of Internet risk, regardless of the constraints that places on their own network use), depending on the visibility and sensitivity of the organization. Special care should be taken to secure websites in order to deny hackers an easy way to embarrass the bureau and to advertise their own causes.

Universities or Colleges

University network administrators have the vexing problem of having to defend their systems from internal attacks as well as external ones. The periodic influx of new students ensures a fresh crop of hackers who will always be pushing at the security boundaries of the network. The students must have computers and access to the Internet, but the administrative staff of the school also needs a secure work environment.

Most schools cope with this problem by having two (or more) separate networks, each with a different security policy and with carefully controlled access between the networks. The public access student network typically has a severely restrictive policy and is frequently checked for viruses, Trojan horses, modified system settings, and so on.

Tip The university or college network administrator usually takes a cautious (deny by default) or a strict (proxy only, no routing) approach to managing the school's administrative networks. The network administrator also takes a fairly open approach to managing the students' network, while taking special care to keep the networks separate and while keeping a close eye on the state of the student network.

Internet Service Providers

The ISP network administrator has a problem similar to that of the university network administrator. The ISP network administrator must keep hackers from the Internet at bay and internal hackers contained, for the customers of the ISP expect to be protected from each other as well as from the outside. In addition, customers expect to have full Internet access—they want to decide for themselves which protocols and services to use.

Tip Most ISPs use a firewall to protect their network service computers (DNS server, mail server, and so on) in a cautious or strict configuration and use a packet filter in a more liberal configuration (permission by default) to stop the most obvious Internet attacks (Ping of Death, source-routed packets, malformed IP and ICMP packets, etc.) from reaching their clients. At the client's request, many ISPs will apply more strict security policies to the client connection on a per-client basis.

Online Commerce Companies

For most companies, the Internet connection is a convenience. For online commerce companies, the reliable operation of the connection and the services that flow over it are the lifeblood of the company. A used bookstore that accepts inquiries for titles over the Internet can afford for its website to be down every once in a while, but an online bookstore that transacts all of its business over the Internet cannot.

In addition to preventing denial-of-service attacks, the administrator of an online commerce network must be aware of a more dire threat—the theft of customer information, including financial transaction data (especially credit card numbers). Consumers expect that the data they provide to your online company will remain confidential, and there may be severe public relations problems if the data gets out, as well as legal repercussions if the company is found negligent in its security precautions.

An online commerce company often has two networks to protect—the internal network used by company employees and another network, perhaps located on the company premises or maybe located at an ISP, that provides the company's online interface to its Internet customers. Each network will have separate security policies; in fact, the online interface must be protected from unauthorized access from the interior network, and vice-versa.

Tip Because of the severe repercussions of both denial-of-service and data-theft attacks, the smart network administrator for an online commerce company will implement a strict (proxy only, no routing) firewall policy for the company's Internet servers. The administrator may establish a more permissive (cautious or concerned) policy for a separate administrative network if the staff needs freer Internet access for business activities that are not business critical.

Financial Institutions

As a general rule, if there is money or there are things worth money flowing over the network, the administrator is going to be particularly careful about who can access the network and how they go about it. The more money there is, the stricter the rules for access will be. Therefore, banks and credit unions never allow any direct Internet access to their financial networks (the ones that directly convey money from one account to another) or even to the administrative networks that bank officials use to perform more mundane tasks.

A growing trend in financial institutions is to allow customers to perform online banking through their web browsers over the Internet. This, of course, means that a web server of some sort must be linked both to the Internet and to the protected financial computers. If you work for a financial institution, you should be sure that every possible measure is taken to secure that web server and protect the customers' account information.

Tip Those banking systems that allow any sort of Internet access implement strict (proxy access only) or paranoid (custom crafted with special purpose network software) policies to protect

their computers.

Hospitals

In a hospital network, unlike all the previous types of networks, people can die if the computers stop working. For this reason, the patient care hospital networks that have medical equipment attached to them are seldom connected to the Internet in any form. Administrative networks may be connected, but those links are carefully secured because of the risk of divulging or destroying confidential patient data. The networks in research labs, however, are typically closely and permissively attached to the network because scientists work best in an open environment where information exchange is made easy.

Tip Like those of banks and universities, the hospital network administrator breaks his networks into several mutually untrusting sections. Life-critical equipment simply is not connected to the Internet. A strict policy is adopted for administrative computers (they still need e-mail, after all), while research LANs have a cautious or concerned policy.

Military Organizations

Military networks, like hospital networks, can have terminal repercussions when security is penetrated. Like governmental bureaus, hackers or espionage agents often have a specific target or axe to grind with the military. But not all military networks are the same—the civilian contractors managing a contract to purchase, warehouse, and distribute machine tools will have a different set of security requirements than the Navy war college's academic network, and neither of those will be designed with anywhere near the level of paranoia that goes into constructing the real-time battle information systems that soldiers use to wage war.

Tip The administrator of a military network must match the firewall policy of the LAN to the type of work performed on it. Classified and administrative networks will have at least a cautious (default deny) or strict (proxy only, no routing) policy, while Secret and above information systems will be divorced from the Internet entirely.

Intelligence Agencies

Some organizations have the dual goals of safeguarding their own networks while simultaneously finding ways to circumvent the walls keeping them out of other people's networks. You can be sure that the professional agents in these organizations have a dossier on and an action plan to exploit every operating system bug or protocol weakness there is. But knowing about a hole and plugging it are two different issues, and sometimes the hackers can steal a march on the spooks.

In an odd turn of fate, the NSA has in fact taken the Linux source code, tightened up security in areas they find important, and released the code back to the free software community. This has given hackers and open-source advocates a bit of indigestion—do you trust it because it is open and you can check the source code, or do you mistrust it because of its source?

Tip It is a good bet that the administrators of these kinds of networks go one step beyond implementation of a strict firewall security—I would be very surprised if these secrecy professionals used any commercial software to firewall their networks. The truly paranoid will only trust software that they personally examine for back doors and weaknesses compiled with similarly inspected software tools.

Configuring the Rules

Once you've determined the degree of paranoia that is justified for your network (or networks if you manage more than one), you can set up the firewalling rules that keep the hackers out. Every firewall allows you to establish a set of rules that will specify what trans–firewall traffic will be allowed and what will not, as well as to establish and manipulate these rules. The following chapters will discuss the specifics of how each firewall is configured.

In the remainder of this chapter, however, you'll learn about these rules generically and how you should establish them so that your firewall won't have any obvious and easily avoidable weaknesses. You'll also learn about the care and feeding of a running firewall and what you can do when you discover it has come under attack.

Rules about Rules

Every firewall worth its weight in foam packing peanuts will have a number of features or characteristics of rules in common. You need to understand these rules and features because they form the building blocks of the logic that will either keep the hackers out or let them in.

Apply in Order

When deciding whether or not to allow a packet to pass the firewall, well–constructed firewall software will start with the first rule in its rule set and proceed toward the last until the packet is either explicitly allowed, explicitly disallowed, or until it reaches the end of the rules (whereupon the packet is allowed or dropped by default). The rules must always be evaluated in the same order to avoid ambiguity about which rule takes precedence.

Some strong firewalls take a "best rule fitting the problem" approach rather than an ordered rule set approach. While this may in fact provide stronger security, it can be very difficult for an administrator to determine which rule will be applied in a specific circumstance.

Per Interface

Firewall software should be able to discriminate between packets by the interface they arrive on and interface they will leave from. This is essential because the firewall can't really trust the source and destination addresses in the packets themselves; those values are easily forged. A packet arriving on an external interface that says it is from inside your network is an obvious flag that something fishy is going on.

Per Type of Packet (TCP, UDP, ICMP)

Your firewall must be able to filter based on packet type because some are essential to network operation, while other types are just recipes for trouble. For example, you will want to allow ICMP echo reply packets to pass into your network from the outside (so your client computers can verify connectivity to outside hosts), but you may not want to pass ICMP echo request packets in to those same clients. After all, there's no sense letting hackers build a list of potential targets on your LAN. Some protocols use UDP on a particular port while others use TCP, and you don't want to let UDP traffic through on a port that has been opened for TCP or vice versa.

Per Source and Destination Addresses

Your firewall must classify traffic according to where it comes from and where it is going. You may want to allow external computers to establish connections to publicly accessible internal or DMZ web and FTP servers, but not to establish connections to internal client computers. You probably want to allow internal clients to establish connections going the other way, however. Your firewall should be able to permanently block troublesome hosts and networks from performing any access at all, and should be able to deny all access to sensitive computers inside your network that don't need Internet connectivity.

Per Source and Destination Ports

Similarly, you will want to control TCP and UDP packets according to which ports they're coming from and going to. You should allow external users to connect from any port on their own computers to just those internal ports that are used by externally visible services (such as HTTP and FTP). Don't allow external users to connect to just any port on internal computers, because Trojan horses such as Back Orifice work by opening up a port above 1023 (most operating systems restrict user programs from opening ports below this value) for hackers to connect to. However, users inside your network need to be able to initiate connections using source ports greater than 1023 with the destination port of any common TCP protocol ports (such as HTTP, FTP, Telnet, and POP). You might want to limit your users to just a few destination ports, or you may allow connections to arbitrary external ports.

Per Options

Originating hosts and routers can set a variety of options in the header of IP packets. Some options are notorious for being used to circumvent security, with source routing as the most abused of all the options. Most firewalls simply drop source-routed packets. Because none of the IP options are required for normal Internet traffic, strong firewalls simply drop any packets that have options set.

Per ICMP Message Type

As mentioned above, some ICMP packets are required for the Internet to cope with network problems. But, many ICMP packets (sometimes the same essential packets) can also be used in unconventional ways to crash computers on your network. The firewall must be able to determine, based on the message type and how it is used, whether or not that ICMP packet is safe to pass.

Per ACK Bit for TCP

The firewall must be able to tell the difference between a packet that is requesting a connection and one that is merely sending or replying over an already established connection. The difference between these two types of packets is just one bit—the ACK bit. Packets requesting a connection have it cleared, all others have it set. You will use this rule characteristic most often with the source and destination characteristics to allow connections to only those ports you specify and in only the direction you allow.

Protocol Specific Proxying Rules

For strong security, packet-filtering rules aren't secure enough. The above packet rules only concern themselves with the header of IP or ICMP packets; the data payload is not inspected. Packet rules won't keep viruses out of e-mail nor will they hide the existence of internal computers. Proxies provide greater security but also limit any ICMP, IP, TCP, or UDP level attacks to the

gateway machine. Proxies also ensure that the data flowing through the firewall actually conforms to the format specified by the protocols that the firewall is proxying for those ports.

Logging

A good firewall will not only block hazardous network traffic but will also tell you when it is doing so, both with alerts and with messages written to a log file. You should be able to log (at your discretion) every packet dropped or passed through the firewall. These logs should be able to grow large enough to track activity over days or weeks, but the logs should never be allowed to grow so large that they fill all of the firewall's hard drive space and crash the computer.

The alert mechanism should not only pop up windows on the firewall's console but also send e-mail to an arbitrary address (such as your pager e-mail gateway, if you are really serious about responding quickly to network attacks and you don't mind those occasional midnight false alarms).

Graphical User Interface

While not necessary for firewall security or performance, a graphical user interface for manipulating rule sets makes it much easier to set up and configure firewalls.

Rules for Security Levels

We've divided the spectrum of security into five levels that will be a good fit for most organizations. Using the first half of this chapter, you should be able to identify which of these levels applies most closely to your organization. Once you've matched your organization to one of the following security levels, you can use the rules we lay out as a starting point for your firewall policy. The general levels are as follows:

- Aware
- Concerned
- Cautious
- Strict
- Paranoid

For each security level we'll explore the rules, restrictions, and procedures that a network administrator will enact to provide that level of security in the network.

Aware

There are some things every security network administrator should do regardless of the degree of security warranted by the network contents or the type of organization the network serves. These actions and prescriptions plug obvious security holes and have no adverse affect on Internet accessibility. The security aware administrator should:

- Install the latest operating system patches on both the client and server computers in the network.
- Keep network user accounts off of Internet service computers such as web servers, FTP servers, and firewalls, and have separate administrative accounts with different passwords for these machines.
- Regularly scan the system logs for failed logon attempts to network services and failed connection attempts to web servers, FTP servers, etc.

- Regularly scan system user accounts for the unauthorized addition or modification of user accounts for network services.
- Disable all unnecessary services on network and Internet servers.
- Use virus scanners on your server (at least).
- Perform regular backups.

Concerned

A network administrator that is concerned about security will at least install a packet filter and take the above "security aware" steps. The packet filter will not stop a concentrated network attack from exploiting service protocol weaknesses, but it will stop the simplest denial-of-service attacks—those based on malformed or maliciously configured ICMP or IP packets.

A packet filter in its most lax configuration allows packets to pass by default unless a rule specifically tells the filter to drop them. Proxy servers may be used to enhance network services (by caching HTML pages, for example) but provide no extra security because network clients can easily bypass them. The packet filter can also lock out troublesome external IP addresses and subnets, as well as deny external access from the outside to specific internal computers such as file and database servers.

Packet Rules (Filtering)

The packet rules control the flow of several different kinds of packets through the filter or firewall. They are as follows:

- ICMP Rules
- IP Rules
- UDP Rules
- TCP Rules

ICMP controls the flow of IP packets through the Internet. IP is therefore essential to the correct operation of the Internet, but ICMP packets can be forged to trick your computers into redirecting their communications, stopping all communication, or even crashing. The following rules (see Table 10.1) protect your LAN from many ICMP attacks

Table 10.1: ICMP Service Rules

Rule	In Interface	Out Interface	Src IP	Src Port	Dest IP	Dest Port	Opt	Ack	Type	ICMP Type	Act
1	*	*	*	*	*	*	*	*	ICMP	Source Quench	Pass
2	*	Ext	*	*	*	*	*	*	ICMP	Echo Request	Pass
3	Ext	*	*	*	*	*	*	*	ICMP	Echo Reply	Pass
4	*	Ext	*	*	*	*	*	*	ICMP	Destination Unreachable	Pass
5	Ext	*	*	*	*	*	*	*	ICMP	Service Unavailable	Pass
6	Ext	*	*	*	*	*	*	*	ICMP	TTL Exceeded	Pass
7	Ext	*	*	*	*	*	*	*	ICMP	Parameter Problem	Pass
8	*	Int	*	*	*	*	*	*	ICMP	Echo Request	Drop

9	Ext	*	*	*	*	*	*	*	ICMP	Redirect	Drop
10	Int	*	*	*	*	*	*	*	ICMP	Echo Reply	Drop
11	Ext	*	*	*	*	*	*	*	ICMP	Destination Unreachable	Drop
12	Ext	*	*	*	*	*	*	*	ICMP	Service Unavailable	Drop
13	Ext	*	*	*	*	*	*	*	ICMP	TTL Exceeded	Drop
14	*	*	*	*	*	*	*	*	ICMP	*	Drop

Note that while we assume that the concerned (but not cautious) administrator allows packets to pass the firewall by default, the above rules work both for permit-by-default and deny-by-default configurations. Rules are (or should be) evaluated in order from first to last, and if an ICMP packet is not specifically allowed or denied in Rules 1–13, Rule 14 will cause it to be dropped. A short description of each rule follows.

1. Allow source quench: You want external hosts to be able to tell your client computers when the network is saturated.
2. Allow echo request outbound: You want your clients to be able to ping external computers to verify connectivity.
3. Allow echo reply inbound: You want your clients to be able to hear the reply of pinged hosts.
4. Allow destination unreachable inbound: These packets inform your clients that an exterior resource is not available.
5. Allow service unavailable inbound: These packets inform your clients that an exterior resource is not available.
6. Allow TTL exceeded inbound: These packets inform your clients that an exterior resource is too far away.
7. Allow parameter problem: These packets inform your clients that they are not sending correctly formatted packets.
8. Drop echo request inbound: Echo request ("ping") packets can be used to survey your internal network for computers to attack. Malformed ping packets are also often sent in an attempt to crash computers.
9. Drop redirect inbound (log it instead): You don't want external agents to be able to mess with your internal computers' routing tables without your supervision.
10. Drop echo reply outbound: Why make it easy for hackers to find computers to attack?
11. Drop destination unreachable outbound: You should protect the identity of interior networks just as you would the identity of individual interior hosts.
12. Drop service unavailable outbound: It is not a good idea to advertise to network attackers what services are available inside your network.
13. Drop TTL exceeded outbound: Hackers can determine the number of hops or LAN boundaries exist inside your network by trying different TTL values.
14. Drop all other ICMP packets: Just to be safe. The ICMP packets you have specifically allowed should be sufficient.

IP There are some rules that you will want to configure for all IP packets regardless of whether they contain TCP or UDP traffic inside them. See Table 10.2 for an overview of the rules, then read further for an explanation of each.

Table 10.2: IP Service Rules

Rule	In Interface	Out Interface	Src IP	Src Port	Dest IP	Dest Port	Opt	Ack	Type	ICMP Type	Act

15	*	*	*	*	*	*	Source Route	*	*	*	Drop
16	Ext	*	Internal	*	*	*	*	*	*	*	Drop
17	Int	*	*	*	Internal	*	*	*	*	*	Drop
18	Ext	*	*	*	Protected Servers	*	*	*	*	*	Drop
19	Ext	*	*	*	*	RIP OSPF	*	*	*	*	Drop
20	*	*	192.168.0.1	*	*	*	*	*	*	*	Drop
21	*	*	Internal	*	Internal	*	*	*	*	*	Drop

These rules govern the general flow of IP traffic in and out of your network. If you want to forbid certain computers or networks from accessing your computers, you would place a rule there, as in rule 20.

15. Drop all source–routed packets: Nobody redirects packets around damaged connections anymore, they just fix the broken equipment. A source–routed packet is a red flag indicating a network intrusion attempt.
16. Drop all packets arriving on the external interface that have a source field indicating that the packet came from inside your network: If the packet claims it originated from inside your network, it should only come from the internal interface.
17. Drop all packets arriving on the internal interface that have a source field indicating that the packet came from outside your network: As in Rule 16, this behavior is an indicator that there's something fishy going on, and you're safest if you just drop the packet and log the event.
18. Drop incoming packets to interior computers that have no externally accessible services (file server, etc.): People outside your network have no business connecting to your LAN file server unless they do it through secure encrypted tunnels that you specifically set up for them.
19. Drop RIP, OSPF, and other router information exchange protocols: You don't need external agents reconfiguring your routers for you.
20. Drop packets from a specific host: This is where you put rules banning hosts and networks that have been bugging you.
21. Drop packets going both to and from internal addresses. These packets shouldn't be on the router in the first place. If they are, it's because some sort of hacking attempt is occurring.

UDP Once you have set rules for generic IP traffic, you will want to further specify some UDP rules to block egregious security holes, such as X–Windows, as shown in Table 10.3

Table 10.3: UDP Service Rules

Rule	In Interface	Out Interface	Src IP	Src Port	Dest IP	Dest Port	Opt	Ack	Type	ICMP Type	Act
22	*	*	*	*	*	0–20	*	*	UDP	*	Drop
23	*	*	*	*	*	6000–6003	*	*	UDP	*	Drop
24	*	*	*	*	*	161–162	*	*	UDP	*	Drop

Note By default, the concerned–but–not–cautious administrator allows packets to pass through the firewall, unless a rule specifically denies that kind of packet.

Each rule specifically denies a port or a range of ports (some firewalls require you to repeat the rule for each port specifically disallowed).

22. Drop packets using ports below 21: There are no services below port 21 that your average Internet user will find helpful.
23. Drop X–Windows (packets using ports 6000–6003): You don't want a hacker gaining control of your mouse and keyboard, do you?
24. Drop SNMP (packets using ports 161 and 162): You don't want network intruders to reconfigure your hubs and routers using this protocol either.

TCP The TCP rules you create are like the UDP rules with one difference—you can use the ACK bit of a packet to stop connections from being initiated from one direction or the other. Blocking inbound packets with the ACK bit cleared (C) for a particular port allows only outbound connections to be initiated, but allows subsequent data traffic for that connection—all of which will have the ACK bit set (S). See Table 10.4 for a typical set of TCP rules.

Table 10.4: TCP Service Rules

Rules	In Interface	Out Interface	Src IP	Src Port	Dest IP	Dest Port	Opt	Ack	Type	ICMP Type	Act
25	*	*	*	*	*	0–20	*	*	TCP	*	Drop
26	*	*	*	*	*	6000–6003	*	*	TCP	*	Drop
27	*	*	*	*	*	161–162	*	*	TCP	*	Drop
28	Ext	*	*	*	*	23	*	*	TCP	*	Drop
29	Ext	*	*	*	*	8080	*	*	TCP	*	Pass
30	Ext	*	*	*	*	1023	*	C	TCP	*	Drop
31	Ext	*	*	*	*	20–21	*	C	TCP	*	Drop
32	Ext	*	*	*	Not SMTP Server	25	*	*	TCP	*	Drop
33	Ext	*	*	*	Not Web Server	80	*	*	TCP	*	Drop

You'll note that some of the TCP ports here are the same as the UDP ports listed in the previous section—some protocols operate over UDP as well as TCP.

25. Drop packets using ports below 20: There are no services below port 20 that your average Internet user will find helpful.
26. Drop X–Windows (packets using ports 6000–6003): You don't want a hacker gaining control of your mouse and keyboard, do you?
27. Drop SNMP (packets using ports 161 and 162): You don't want network intruders to reconfigure your hubs and routers using this protocol either.
28. Disallow incoming Telnet connections (incoming packets with destination port 23): Telnet is an insecure protocol because the account name and password exchange are not encrypted and neither is the data channel once a Telnet session is established. If you need secure exterior console access on Unix systems use SSH (port 22) instead.
29. Specifically allow any internal services that use ports greater than 1023: This way you can use the next rule to stop back door software such as Back Orifice, which opens ports internally for remote unauthorized control of your computers. In the rule base above, we've shown an example of allowing external traffic to a web server running on port 8080. Your custom rules will vary depending upon your needs.

30. Drop SYN packets from outside to internal ports >1023: Most legitimate services are configured on ports <1024 so this rule stops connection requests to ports higher than 1023. Rule 28 and others like it (which must be placed before this rule) specifically allow any exceptions.
31. Disallow incoming FTP data connections thus allowing passive FTP only.
32. Disallow SMTP connections (port 25) from the outside to other than your mail server.
33. Establish service destination rules for other services such as HTTP.

Cautious

Most network administrators feel that the aforementioned rules are not enough; a dedicated hacker with time and resources can find a way around these rules. A cautious network administrator will take a more conservative track and block all traffic by default, only allowing the traffic that seems safe after careful consideration. This approach takes much more time and effort to set up correctly, but the result is a much more secure firewall.

Network Address Translation One feature that cautious administrators really like is Network Address Translation, or NAT. This nifty feature allows you to expose just a handful (or even one) of IP addresses to the outside world, while it hides a whole LAN of tens, hundreds, or even thousands of computers behind it. To the computers on your LAN there is little difference between existing on a NAT hidden LAN and being directly connected to the Internet. The firewall keeps track of connections and rewrites packet source and destination and port values on the fly.

Fragmentation A cautious network administrator won't allow fragmented packets into the network. Until recently, fragmented packets were considered safe because it was understood that only the first fragment of a fragmented packet needed to be examined and dropped if necessary. Any subsequent packet fragments would be dropped by the destination computers because they lacked an initial packet with the header information.

Because of a bug discovered in common TCP/IP protocol implementations (including that of Windows NT), that is no longer the case. The bug (since fixed) ignored improper fragment offset values which allowed the second (or later) fragment to occupy the memory location of the first fragment and to provide the header information the network stack was looking for (header information that had not been checked by the firewall). Although the bug has been fixed, most cautious network administrators choose to reassemble fragmented packets at the firewall or just drop them since the fragmentation feature is largely obsolete.

ICMP, TCP, and UDP Some of the packet rules listed in the previous sections become redundant when the cautious administrator denies all packets by default. You may want to leave them in your rule set so you can switch from deny-by-default to allow-by-default and back again when you are diagnosing network connectivity problems.

One rule that is not listed here (because it is denied by default) is that you shouldn't allow Telnet connections (connections to port 23) to travel through your firewall at all. Use the Secure Shell (SSH), instead because it provides much greater security for a remote terminal. In addition, we don't explicitly list the rule you should block access out to ports above 1023 because the commonly accessed services live below 1023. See Table 10.5 for a few additional rules that the cautious administrator will want to configure in the firewall.

Table 10.5: The Additional Rules of a Cautious Administrator

Rules			Src IP		Dest IP		Opt	Ack	Type		Act

	In Interface	Out Interface		Src Port		Dest Port				ICMP Type	
34	*	*	SMTP Server	*	*	25	*	*	TCP	*	Pass
35	*	*	NNTP Server	*	Ext NNTP Server	119	*	*	TCP	*	Pass
36	*	*	Ext. NNTP Server	119	NNTP Server	*	*	*	TCP	*	Pass
37	Int	*	*	>1023	*	80	*	*	TCP	*	Pass
38	Ext	*	*	80	*	>1023	*	S	TCP	*	Pass

- 34. Allow outgoing SMTP connections (port 25) from your mail server.
- 35. Allow NNTP from your news server to external news server (Port 119)
- 36. Allow NNTP from external news server to internal news server (Port 119).
- 37. Allow clients to establish HTTP connections (source port >1023 destination port 80).
- 38. Allow data traffic from an already established HTTP connection to travel back from port 80 on the Internet to the client port (>1023) on your LAN. It is usually not necessary to specify return channel rules on modern stateful firewalls.

You can repeat Rules 36 and 37 for any additional Internet services that you want internal clients to be able to connect to.

Service Rules (Proxying)

General rule: Disallow proxy requests from the Internet by filtering out packets with the ACK bit clear that connect to your proxy from the external interface. This prevents hackers from connecting to your proxy server and using it to launder their connections.

Proxy DNS using a DNS server or a proxy server that supports DNS proxy transparently, but disallow zone transfers. You don't want anyone outside your network reconfiguring your network names.

Use SOCKS to proxy stream-oriented (circuit level) TCP and UDP going out of your network for protocols you intend to allow that do not have specific security proxy software.

Proxy SMTP by having your mail servers write e-mail messages to disk and having another mail process read and forward those messages inside the network. This prevents any single process from being compromised, which would allow a connection all the way through the machine. It also guarantees that any malformed e-mail will either crash the inbound process and be dropped or will be completely regenerated by the forwarding process without the deformity. This prevents buffer overrun conditions on mail servers inside your network.

Proxy NNTP by establishing an NNTP server in your demilitarized zone. Inbound NNTP should be forwarded only to that machine, and internal newsreaders should only be able to read news at that machine. This prevents Trojan horses or other non-NNTP traffic from exploiting the open NNTP port on the packet filter.

Proxy FTP using a dedicated FTP proxy application in the firewall. FTP service filters on stateful firewalls may also be used as long as they are specific to the FTP protocol.

Proxy HTTP through a secure HTTP proxy. Your HTTP proxy should be capable of virus checking and stripping executable content.

Disable SOCKS proxies because of the wide protocol access SOCKS allow.

Do not provide POP access at your exterior border gateway. Your SMTP servers should be inside your DMZ. For remote users, force the establishment of an encrypted tunnel for mail access. This prevents e-mail from being exposed in an unencrypted form on the Internet. Internal users who wish to access their private POP accounts should use one of the many free gateway services available on the Internet so they can use their web browser to check their private e-mail.

Do not provide a Telnet proxy. Telnet is far too insecure to allow public access through your gateway.

Strict

The basic strategy for strict control is to completely disallow Network-layer routing between the public and private networks. Strict policies use only Application layer proxies (and not many of them) to provide only the most useful Internet services. Stateful packet filters should be used to protect the application proxies on the bastion host, which should perform no packet routing.

The bastion host may perform Network Address Translation to make itself "invisible" to internal hosts so that client applications don't have to be explicitly set to use a proxy. In this mode, the proxy host looks like a router to internal hosts; it receives all traffic as if it were going to forward it directly onto the public network. That traffic is redirected to proxy applications on the bastion host instead of being routed, so the connections are in fact regenerated to eliminate any undetected deformities. This mode also makes it easy to proxy difficult protocols like FTP and H.323.

Packet Rules (Filtering)

No routing should exist between the external public and internal private interfaces of the bastion host. The same ICMP and IP rules described for the previous section should be applied to protect the bastion host from denial-of-service attacks.

Service Rules (Proxying)

The basic strategy behind strict security is to proxy only the most useful protocols: HTTP and SMTP. These two protocols, which are easy to control and keep track of, allow most of the functionality of the Internet to be utilized. Your attack risk increases linearly with the number of special cases you define on your bastion host. So, the policy here is to limit to the utmost practical degree.

Proxy DNS using a DNS server, but disallow zone transfers. You don't want anyone outside your network reconfiguring your network names. Make sure your DNS proxy does not publish internal names.

Proxy HTTP through a secure HTTP proxy, like NAI Gauntlet. Strip all executable content including EXE, ActiveX, and Java applets. Consider putting a client in the DMZ where these restrictions don't apply if you need to download executable attachments or visit a site that requires Java or ActiveX.

Proxy SMTP by having your mail servers write e-mail messages to disk and another mail process read and forward those messages inside the network. Configure your mail server to strip all

executable attachments and perform virus checking on ZIP compressed and office documents. The only allowable method to transfer executable content through the firewall should be in a non-executable form like BIN-HEX or compressed format. This prevents users from clicking on Trojan horse attachments and executing them. The extra level of indirection ensures that they will at least manually inspect the files before running them.

Paranoid

The strategy for paranoid installations is to either not connect to the Internet or connect a separate "Internet only" network to the Internet. The U.S. government does both to protect its classified networks from the Internet: they are completely disconnected, and a separate pool of "disposable" machines are available for users to work on the Internet. You should simply use a pool of dedicated Internet machines behind a standard stateful inspection filter.

Packet Rules (Filtering)

Allow no direct routing between the public and private networks.

Service Rules (Proxying)

Allow no proxying between the public and private networks.

Case Study: All The King's Firewalls

As I wrote this chapter, I agonized over the case study: What could I talk about that would justify the "paranoid" strategy? Reality intruded on my behalf. While this chapter was being written, the Code Red (cross platform!) buffer-overflow worm appeared and passed unscathed through even the most rigorous firewalls in the world, wreaking havoc and causing uncountable millions of dollars in damages.

Worms are malicious programs that exhibit characteristics of both Trojan horses and viruses. In fact, the Nimda virus, another recent Internet plague was both a worm and a Trojan horse that propagated using the methods of a virus that required user activation.

An e-mail worm typically works like this:

An e-mail message appears in the victim's inbox containing the message text "I received your e-mail and shall send you a reply ASAP. Till then, take a look at the attached zip docs." The sender's address would be the valid address of a close friend or coworker. The attached executable file would be called `zipped_files.exe`.

Upon clicking the attachment, the Trojan horse functionality is activated. The worm creates e-mail messages to everyone in the user's personal address book, thus propagating itself to all of them and appearing to come from the activating user. The worm also rifles through the shared directory structure on the computer and propagates itself to other machines on the local network.

The worm then goes on a rampage, destroying programming code, office documents, and other useful work-related materials, as well as modifying system files on the computers.

Because the virus is attached to e-mail and transmitted by the users who have privileged access through the firewall, it passes through even the strongest firewalls unmolested. Only those organizations whose administrators have insisted on stripping executable attachments from e-mail

are safe.

The only way to truly protect an organization from Internet-based attacks is not to connect to the Internet. Using the paranoid model discussed above is protection against even the most devious schemes.

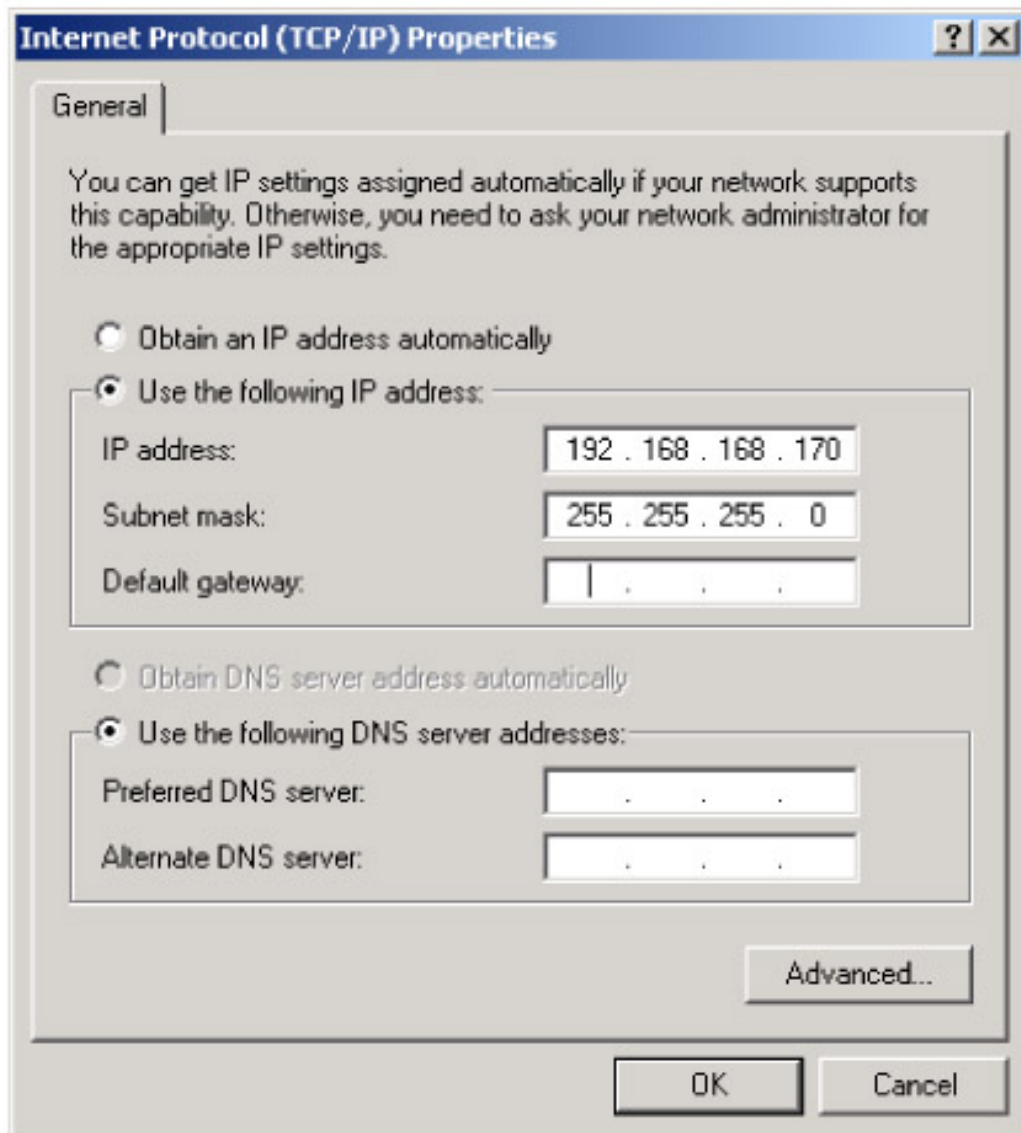
Chapter 11: Configuring a Real Firewall

This chapter is a visual tour through the configuration of a SonicWALL Pro VX, a powerful firewall from SonicWALL. As such, it shows every feature that this line of firewalls supports, and this line of firewalls represents the state-of-the-art in device-based firewalls.

This chapter is not a review—the comparative review for SonicWALL devices can be found in Chapter 19. This chapter simply describes the features and configuration of this firewall as a complete introduction in case you've never dealt with a firewall before. If you have done it before, you'll probably just want to skim through this chapter.

The SonicWALL Appliance Wizard

SonicWALL devices come from the factory with the preconfigured IP address 192.168.168.168/24. This means that in order to attach to the device, your management workstation's IP address must be within the 192.168.168 subnet. In Windows 2000, you can simply set your IP address manually and plug the SonicWALL into the same Ethernet network in order to reach it. This graphic shows a management workstation's IP address set to 192.168.168.170 in order to begin the SonicWALL configuration.



After configuring your management workstation's IP address, open a web browser (Internet Explorer or Netscape) and direct it to <http://192.168.168.168>. When you do, the SonicWALL Appliance Wizard screen will appear as shown here.



To avoid the common problem of shipping a device with a standard default password that might never be changed, the SonicWALL Wizard requires you to change the administrative password as the first configuration step, as shown here.

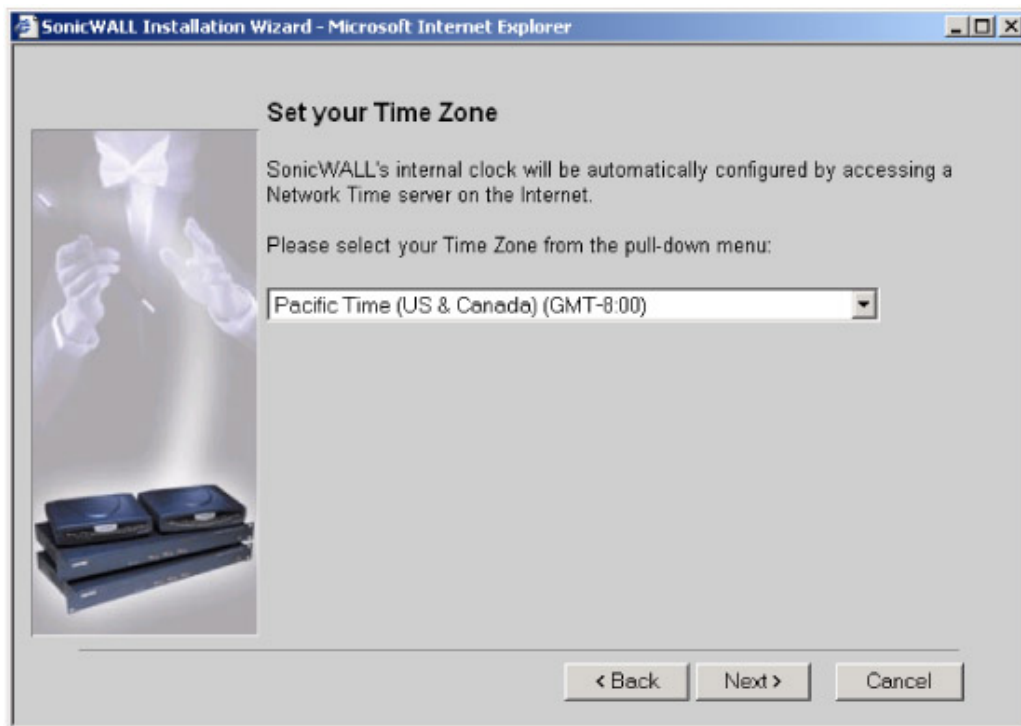


Of course, you should choose the strongest possible password and you should not use the password on any other non-firewall devices to prevent its compromise. Although SonicWALL devices can only be configured from the LAN port or by administrators who have authenticated with the VPN, there are many unobvious ways to get LAN access from outside the network. Opening port-forwards for Terminal Services or VNC, connecting an improperly secured 802.11 wireless bridge, or a user accidentally downloading a Trojan horse are just a few of the ways that a hacker might be able to gain access to the web-based firewall management interface from the interior of

the network.

In fact, I frequently set up a temporary port-forward on new SonicWALLs to facilitate the establishment of a permanent VPN connection. I do this because it allows me to work simultaneously on both devices even though no VPN exists yet. Once the VPN is established, I remove the rule that forwards the dangerous service through to the internal network. You will probably find yourself doing this as well, so be certain you remove these dangerous rules.

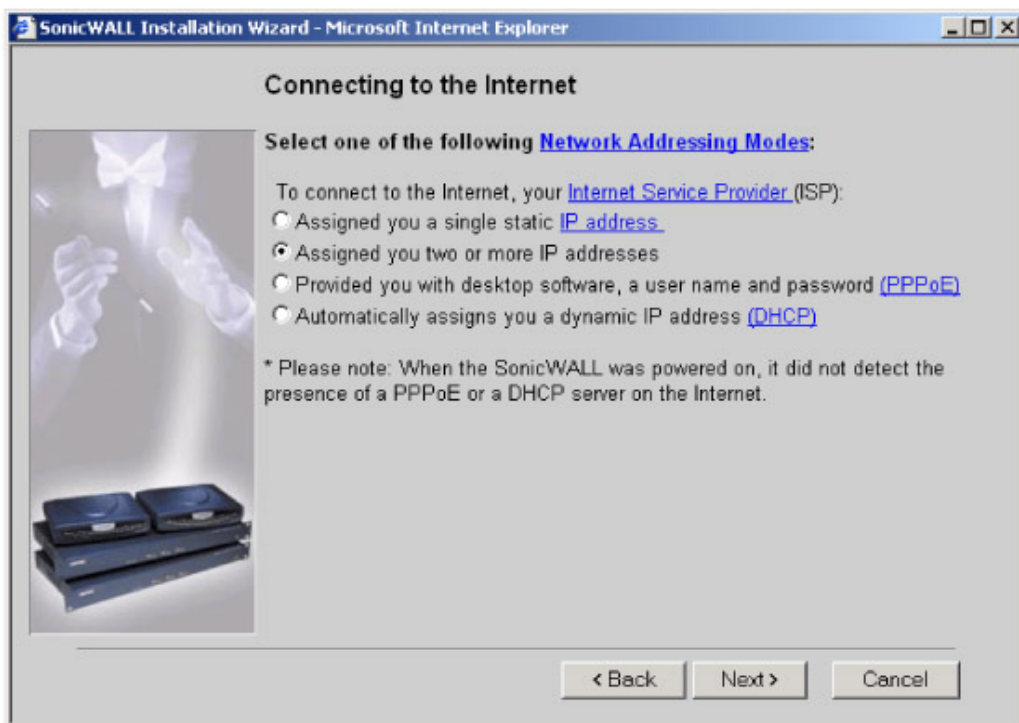
The next step is to set your time zone. SonicWALL devices automatically configure their internal time using the NTP (Network Time Protocol) to synchronize with the Universal Standard Time generated by the U.S. Naval Observatory. You have the option of changing the NTP time server in the administrative interface once the initial configuration is complete. It would be nice if the firewall could be configured as an NTP time server for the rest of the network, but it can't.



The next pane is simply an information pane that asks you to gather your IP circuit network information from your ISP and informs you that the WAN port has not been connected if you have not connected it. You must have this information ready in order to proceed, and you should connect the WAN port now to ensure that everything works correctly during the configuration.

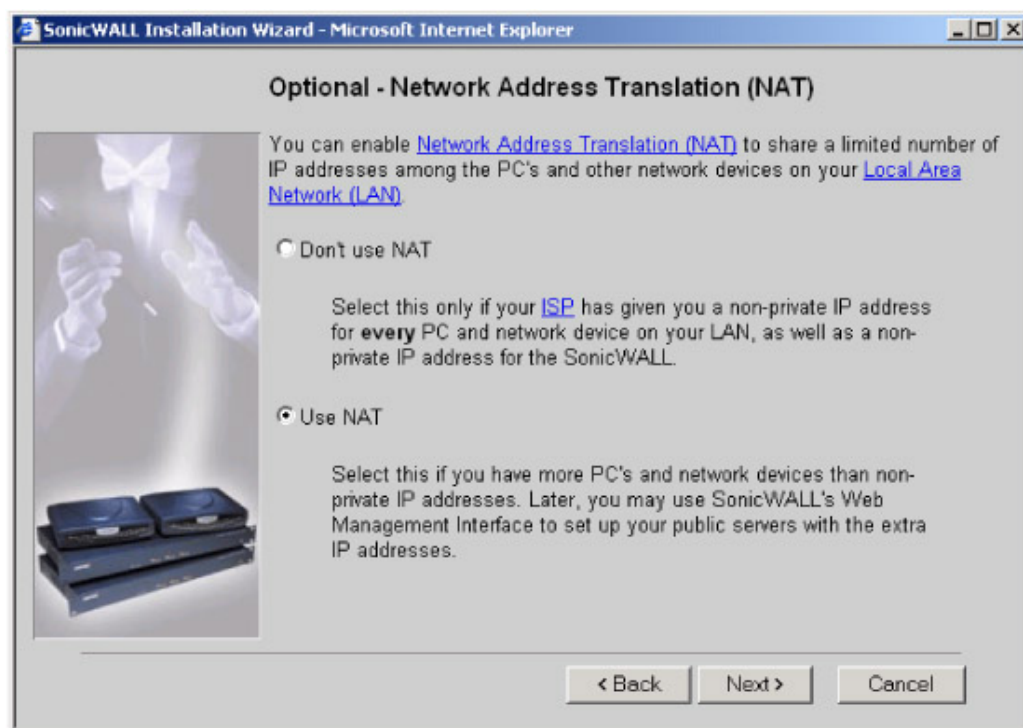


The SonicWALL Configuration Wizard then asks you what type of service you have from your ISP, and makes some assumptions about how you want to configure the device based on answer. For example, if you respond that you've received only a single IP address or that your device receives its address from a PPPoE (Point to Point Protocol over Ethernet) or DHCP server, the wizard assumes that you want to enable NAT. If you indicate that you've received multiple IP addresses, the device will ask you whether you want to enable NAT. In any case, just select the answer that comes closest to your grade of service.



If you've selected the multiple IP address option, the wizard will prompt you whether or not you want Network Address Translation enabled. In the vast majority of cases, you do. Besides conserving your public IP addresses and allowing you to grow your network irrespective of the number of

addresses your ISP has assigned you, Network Address Translation has built-in immunity to a number of hacking attacks. But if you are really certain that you don't want NAT, select the Don't Use NAT option, shown here.



After you've selected your service type and determined whether or not you want to use NAT, the wizard will prompt for your public IP address information provided by your ISP. This graphic shows the configuration for my company's firewall with the actual IP addresses blanked out. Normally your IP address would appear.



After configuring the public IP address information, you will enter your private IP address information. You can choose any IP range you want, but you should never vary from using a

reserved IP block like the 10.0.0.0/8 range or the 192.168.0.0/16 range because if you do, you'll prevent your users from reaching any public IP services with coincidental IP addresses. You may want to avoid using the 10 range as well because it's used for internal routing by a number of second-tier ISPs and could potentially cause conflicts for you. If you have strange routing problems using the 10 reserved block, contact your ISP to determine if they're using any portion of it.. Here you can see the internal IP address configuration for a firewall configured to use NAT.



SonicWALL Installation Wizard - Microsoft Internet Explorer

Fill in information about your LAN

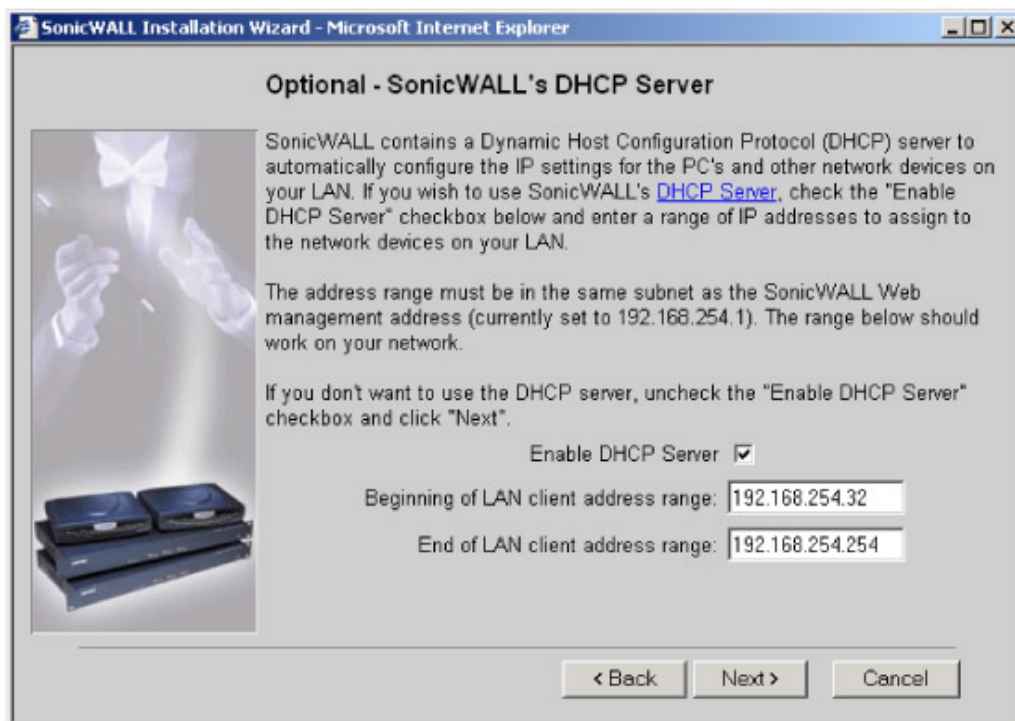
Since you are using [NAT](#) , you will need to specify information about your [LAN](#). You can choose this information arbitrarily, but it's a good idea to use "private" addresses (such as 10.0.0.1 or 192.168.254.1). Note that the default values below will work well for most networks.

SonicWALL LAN [IP Address](#): 192.168.254.1

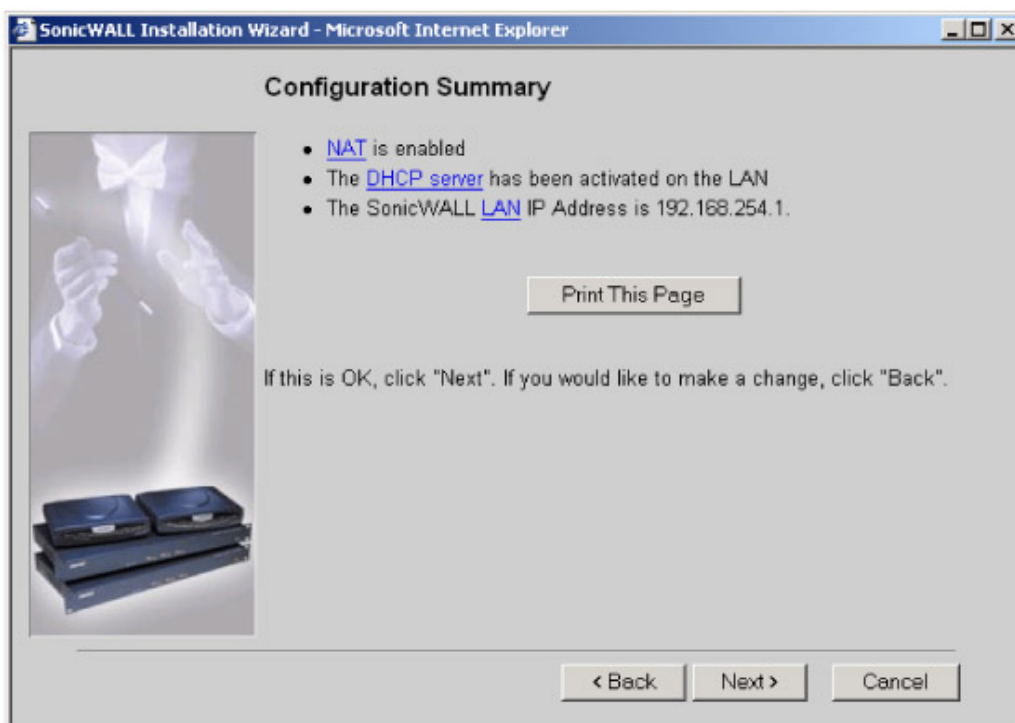
LAN [Subnet Mask](#): 255.255.255.0

< Back Next > Cancel

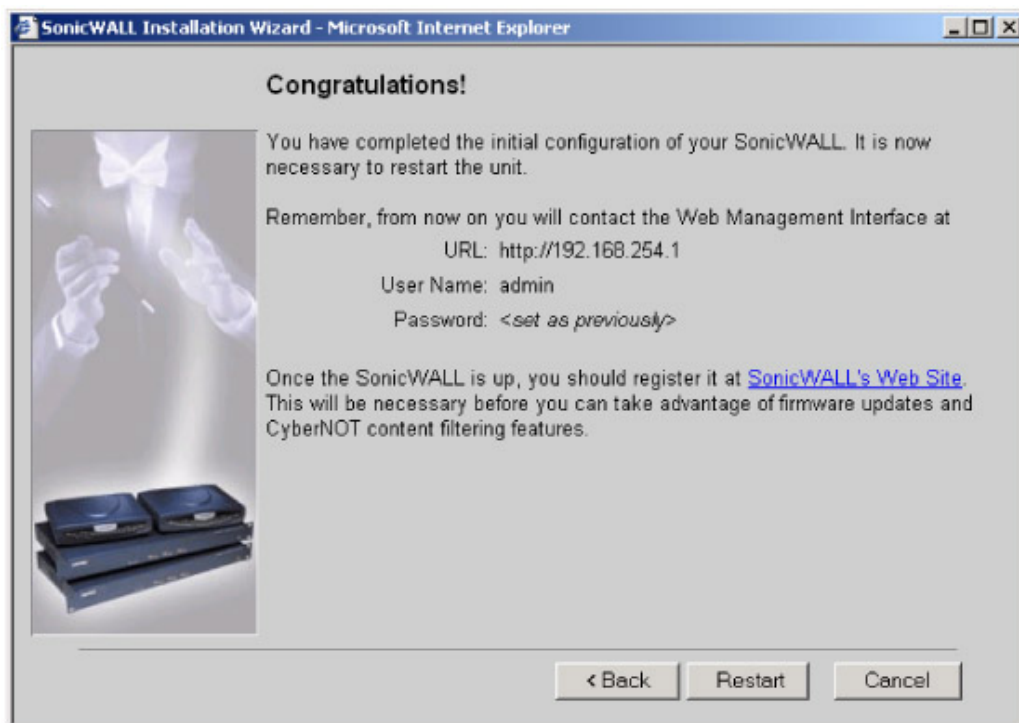
The next pane asks you whether you want the SonicWALL device to provide DHCP addresses, and if so, what range you want DHCP to assign. I generally configure DHCP to be served by a solid-state device in the networks I manage because it's somewhat more reliable than using general-purpose servers to provide DHCP. However, servers are usually a little more flexible and easier to configure if you need to use a large number of static DHCP entries. In my experience, it's easiest to permanently assign static IP addresses to every device that will provide any sort of service, and then use DHCP for clients that provide no services. In this environment, firewalls make ideal DHCP servers.



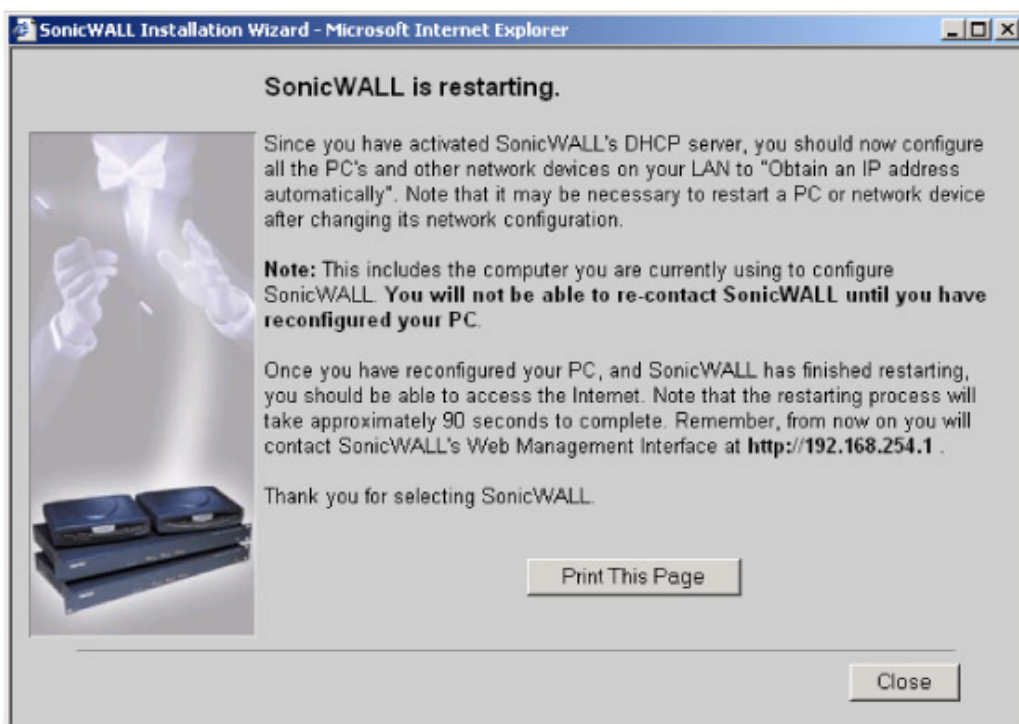
After you've configured DHCP, the firewall initialization is complete and the wizard shows a summary page that looks like this.



The final step in the configuration is to reboot the firewall. Once you do this, the firewall will come up on its new IP address, so you'll no longer be able to reach it until you reconfigure the IP address on your management workstation. Before clicking Restart, you can page back through the Wizard to check your settings and change anything that was entered incorrectly. It's crucial that you remember what the SonicWALL's NAT IP address is and that it is set correctly, because even if you flash the SonicWALL's firmware and reload its operating system from a binary image, the NAT address will not revert to 192.168.168.168. You'll have to use a sniffer and an ARP tool to determine its IP address if you've forgotten it. Here is the restart pane of the SonicWALL Configuration Wizard.



After you click restart, the SonicWALL Wizard displays an informational pane to keep you occupied for the 30 seconds it takes the device to restart.

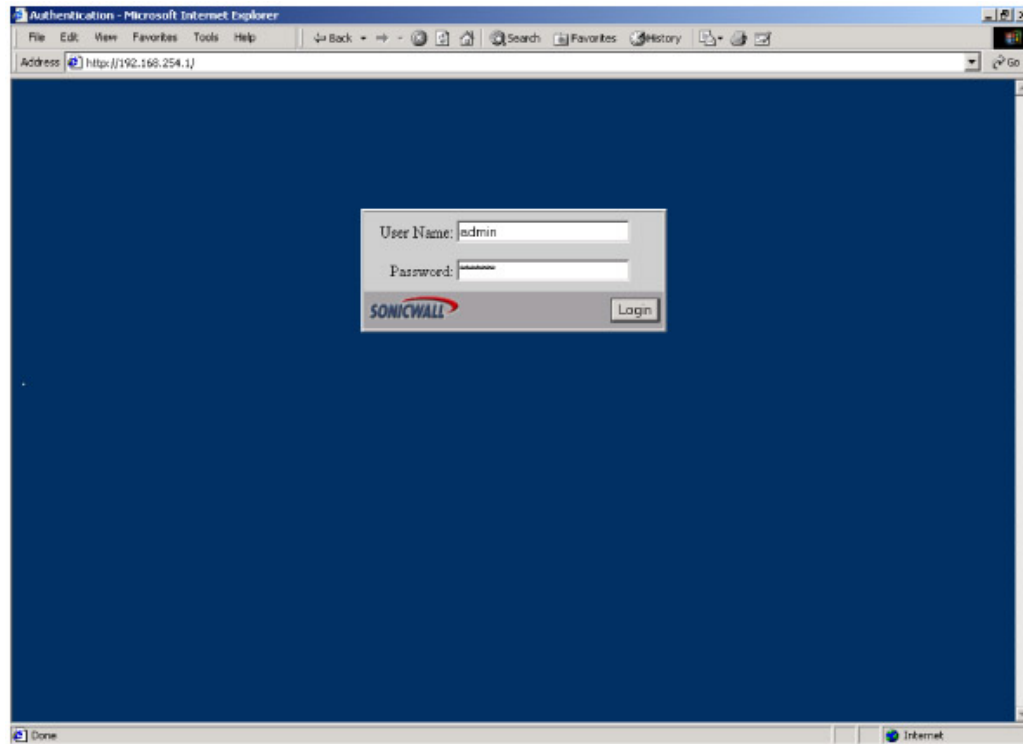


At this point, you've completed the SonicWALL Wizard and the firewall is ready for basic operation. To continue configuring the firewall, you'll use the built-in web management interface.

SonicWALL Registration

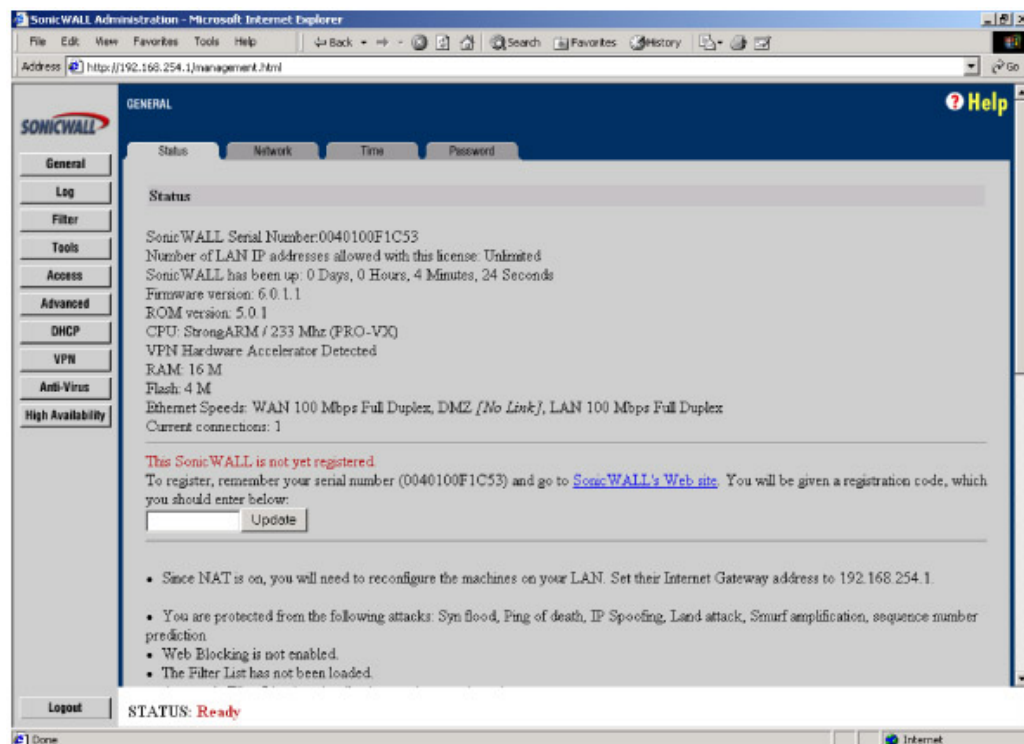
Once you've completed the Configuration Wizard and restarted the firewall, point your web browser

back to the SonicWALL's LAN IP address. This time, instead of the Configuration Wizard you'll get the SonicWALL Web Manager login prompt.

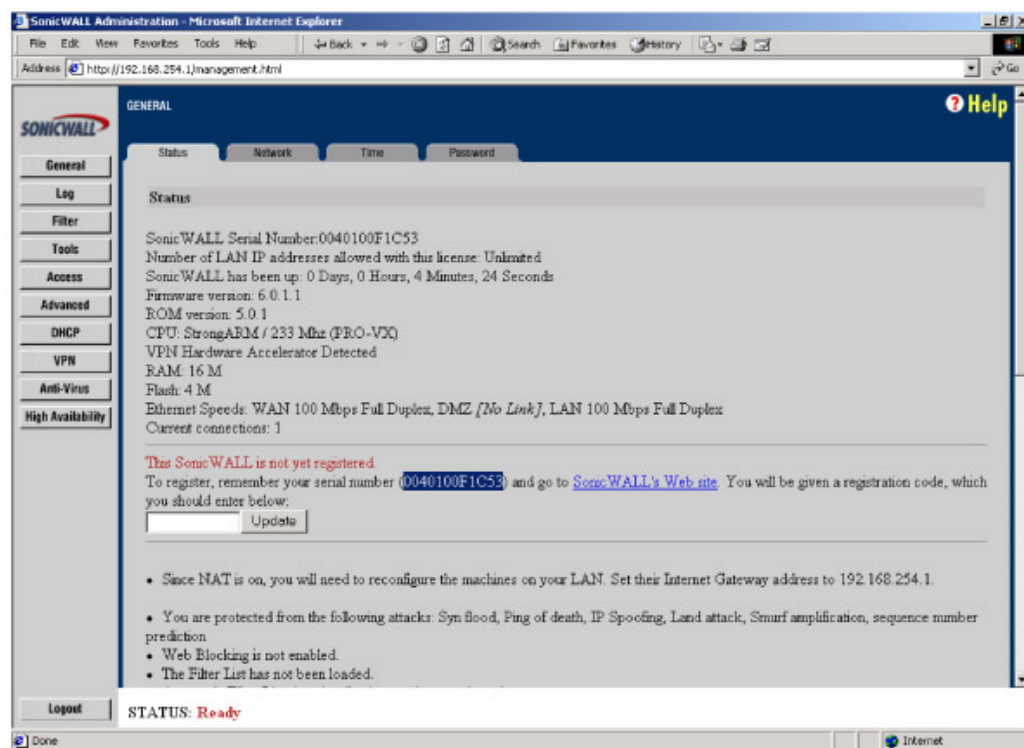


If you're using Internet Explorer, the login prompt can be confusing because it won't actually work until the Java applet (the applet is that tiny grey dot on the right side of the screen) used by the device actually loads completely. This is indicated in the web browser status bar in the lower left-hand corner of the screen. Until the status bar says "Done," any attempts to log in will be responded to with a JavaScript error. Unless you know this, attempting to log in can be frustrating. Netscape does not exhibit this problem because it's a side-effect of the way that Internet Explorer supports Java.

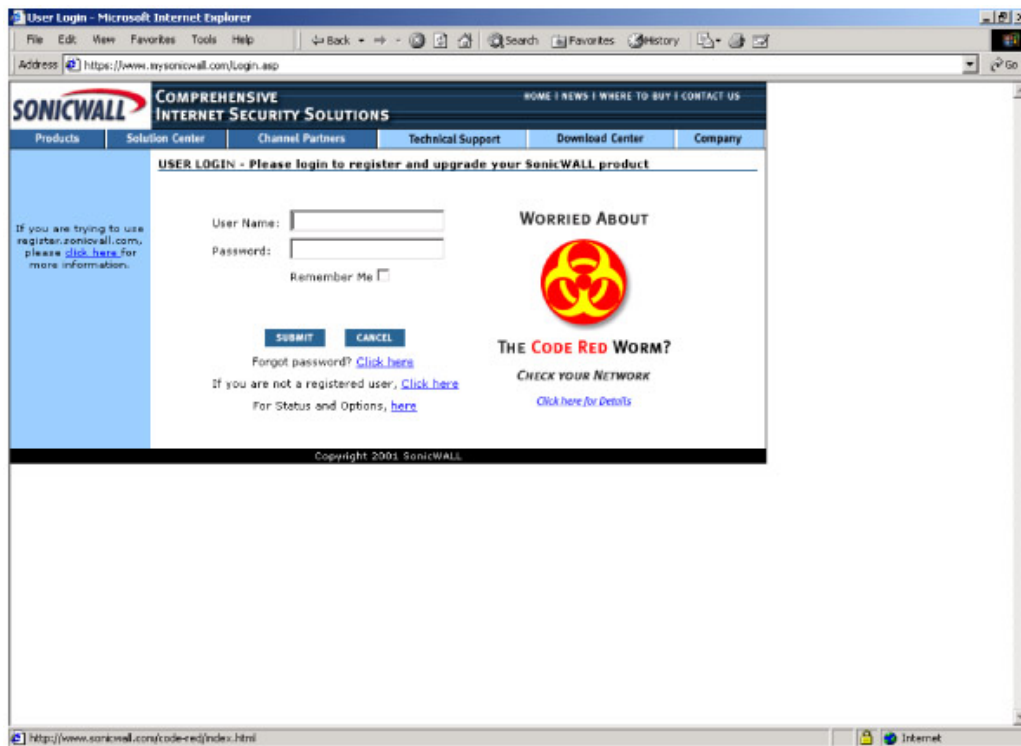
Once you've logged in, you'll see the General Status page. This page provides low-level status of the SonicWALL device, along with support information. Any internal problems the device is experiencing will show up as red text in the General Status page—for example, if the DMZ or WAN interfaces are not connected or if the SonicWALL is not registered.



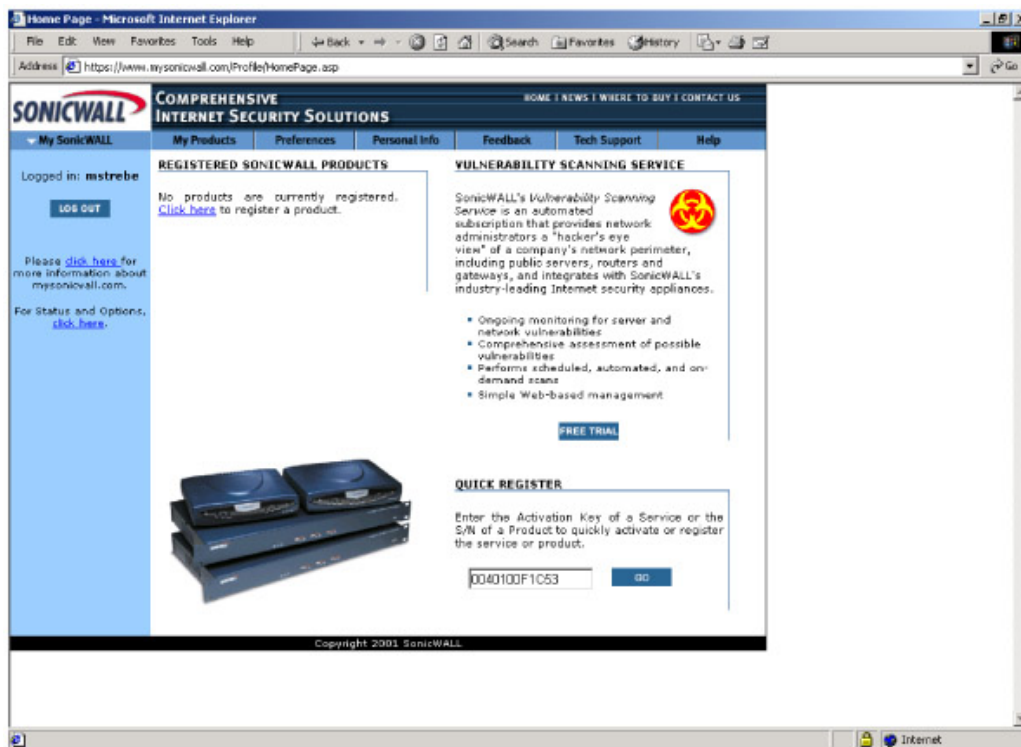
The first thing you should do (though it's not required for the device to operate) is register your firewall. This activates your support and warranty services and allows you to purchase additional services such as VPN (if it wasn't included in your device), website filtering, virus scanning, and so forth. The easy way to do this is to select your firewall's MAC address (which serves as the serial number) in the general page and copy it to the clipboard.



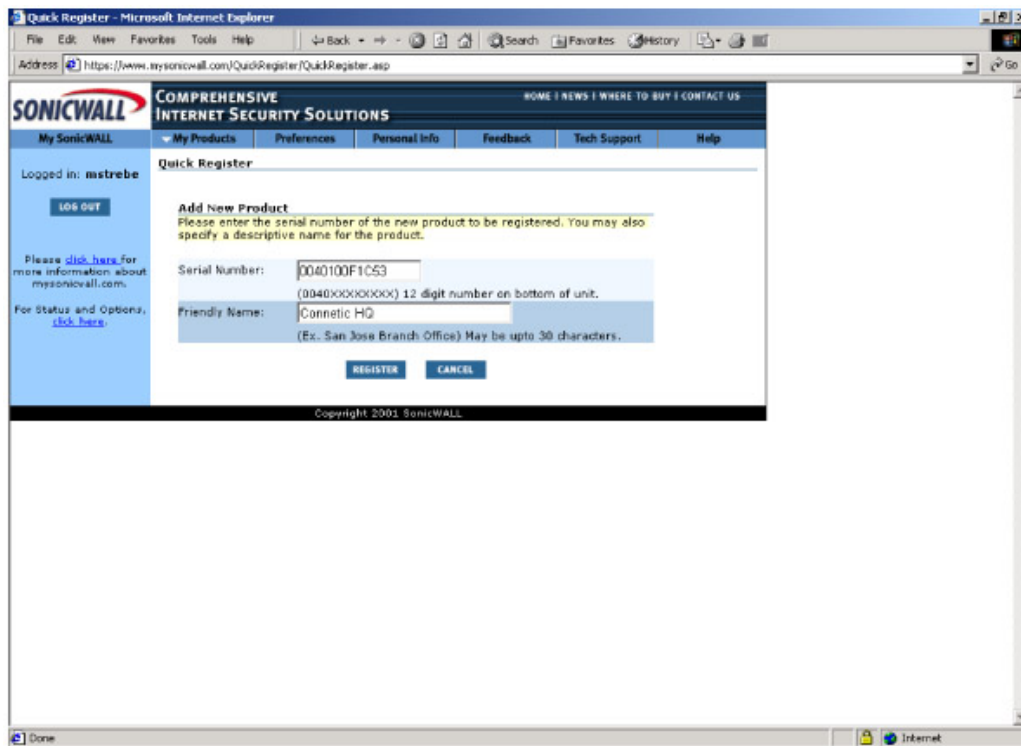
Then you can register your firewall by going to <http://www.mysonicwall.com/>, creating an account for yourself, and logging in. Here is the SonicWALL website login prompt.



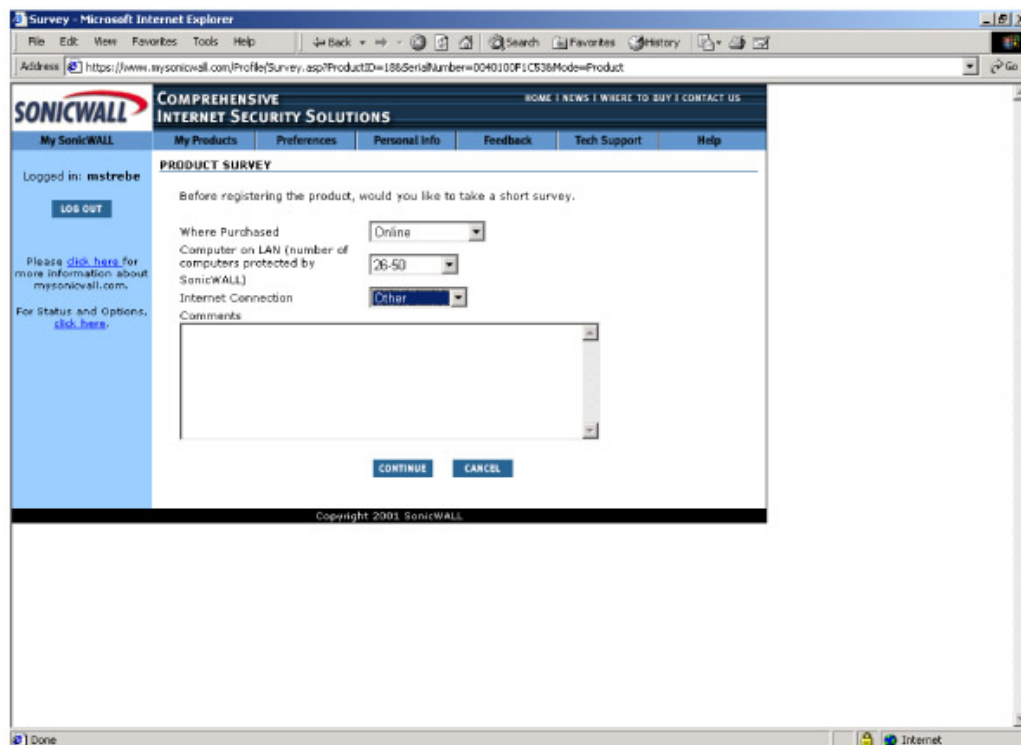
Once you've logged in, you'll see the registered products page. All the SonicWALL devices registered to your account will appear on this page, shown here.



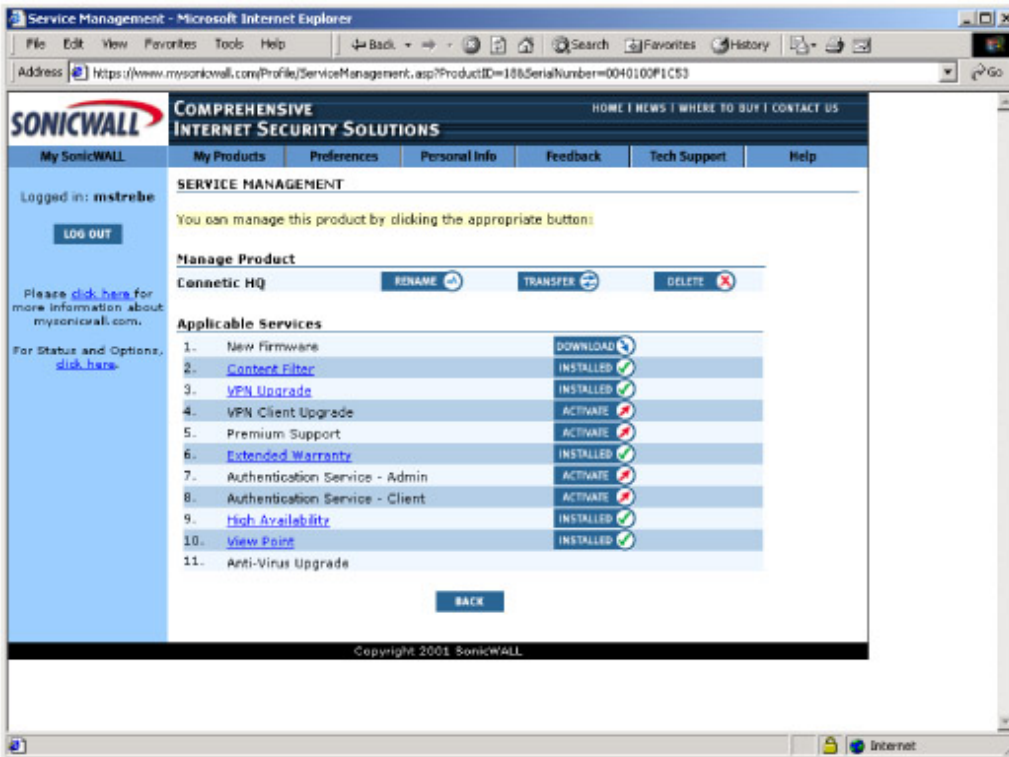
Click the product registration link to bring up the Quick Registration page shown in next. You'll enter the device's MAC address (which is also the serial number) and a descriptive name in order to register the product to your login account.



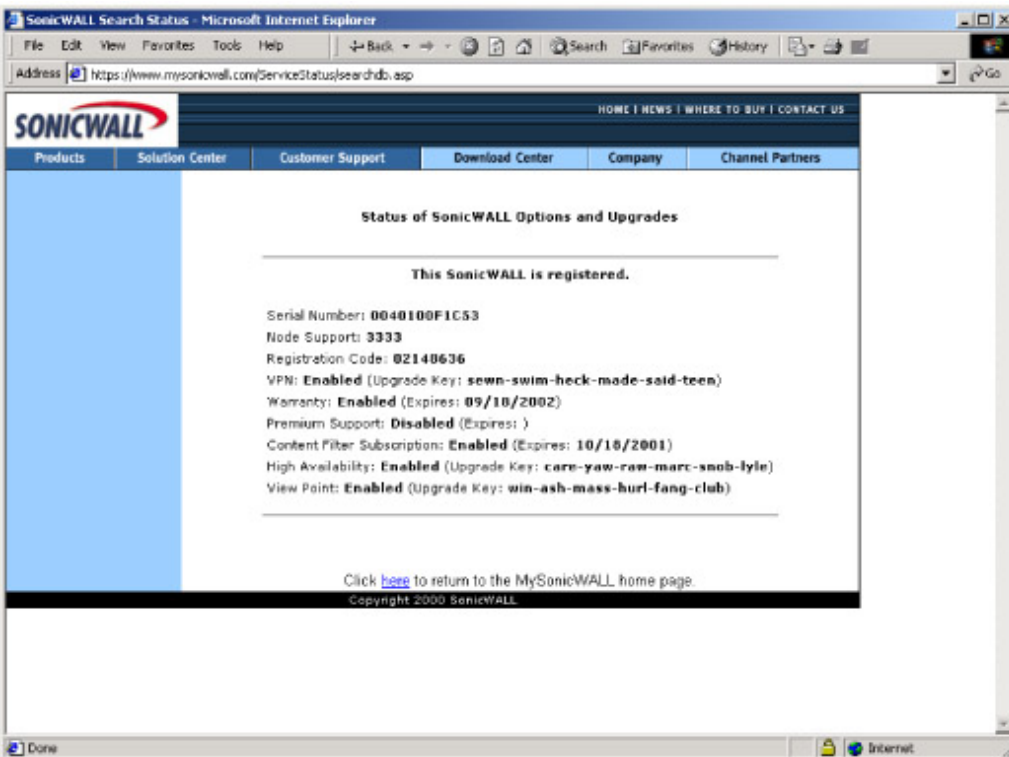
Of course an annoying marketing survey pops up next, as shown next. Just click Continue to dismiss it, and then click the newly registered firewall in the Registered Device page.



The next screen shows the SonicWALL Service Management page, which provides a convenient place to access the latest firmware upgrades for your device as well as keep track of your registration codes for value-added services that you subscribe to. You'll need to enter these codes in the device to activate various features, and they'll be needed again if you ever have to clear the firewall.



The Registration Status page shows the registration status and activation codes for a specific firewall device, as shown next. You'll need to copy the registration code on this page and paste it into the firewall's general status page in order to clear the unregistered error condition on your firewall's General Status page.



You'll also visit this page whenever you need to activate a new service or reactivate a service after clearing a firewall's configuration; this is rare, but sometimes required in certain troubleshooting operations.

SonicWALL Configuration

Once your firewall is registered, you're ready to begin configuring the firewall with your specific Internet access policy. As explained elsewhere in this book, you should take a pessimistic approach to security by disabling all access and then specifically enabling only those protocols your organization requires to perform its work.

SonicWALL device configuration is divided into topical sections, which form a column of navigation buttons on the left side of the screen. The General Status page, shown in the next section, appears as it would after the firewall has been registered. For each topical section, a row of tabs displayed across the top indicates each configuration detail page.

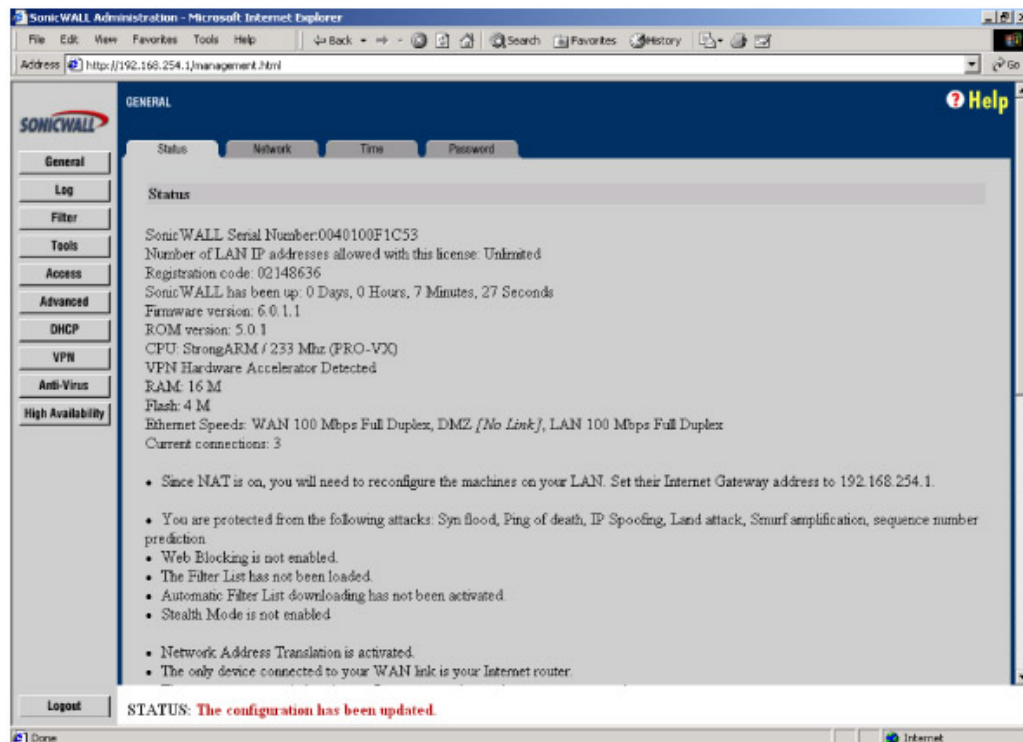
Context-sensitive help is available anywhere by clicking the Help icon in the upper right-hand corner. The help pages are essentially the product documentation in HTML format. When you click the icon, a new browser window will appear with the specific section of the documentation for the current page showing.

This layout is clear and easy to navigate, and it is a major reason why these firewalls are so popular. In the following sections, we'll discuss each of the navigation buttons, and the pages available under each topic.

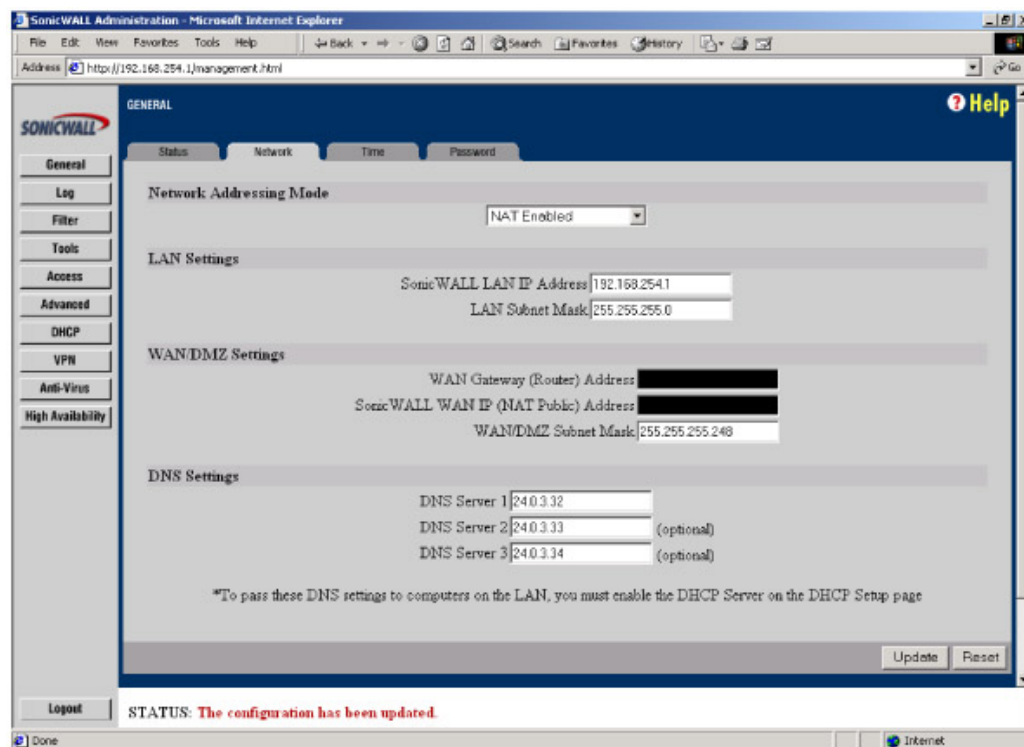
General

The General topical section is used for low-level network configuration of the firewall. All of the information in this section is automatically filled in by the Configuration Wizard that you've already run, so you won't need to configure anything on these pages initially—unless you need to correct a mistake made in the Configuration Wizard.

The Status page shown here shows the hardware status of the device.

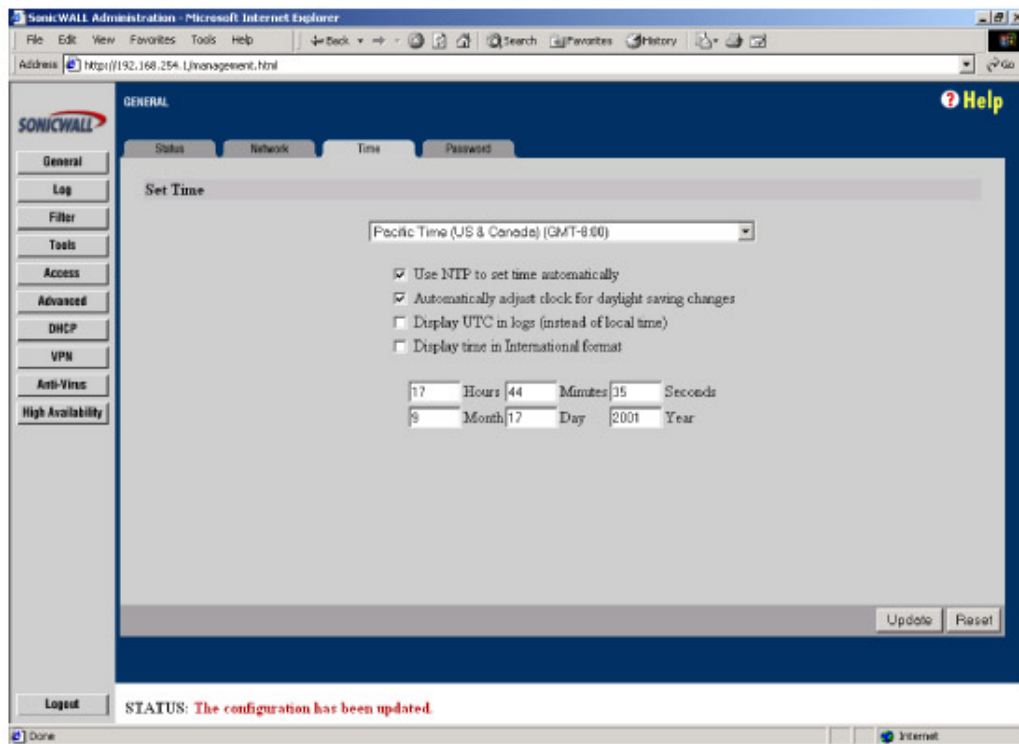


The General Network page allows you configure the device's WAN and LAN IP addresses and enable or disable NAT. You can also enter DNS servers for the firewall to use when accessing public services like mail and time servers for the operation of the device.

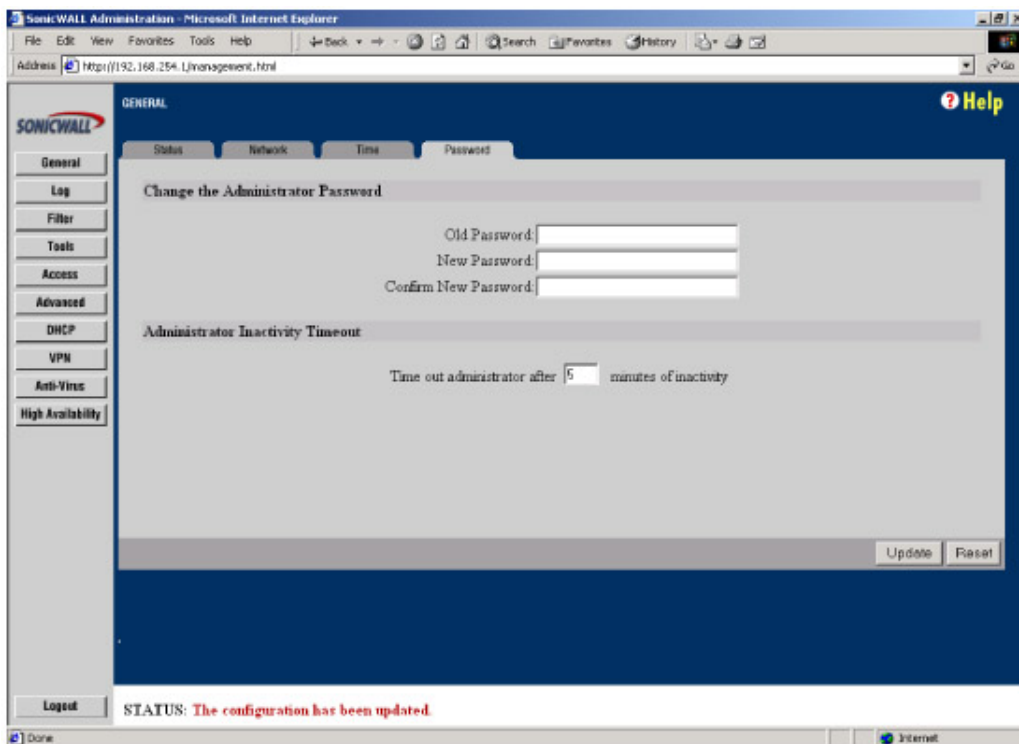


The terminology used on the General Network Page can be slightly confusing. The WAN Gateway (Router) Address is the default gateway provided for your use by your ISP. The SonicWALL WAN IP (NAT Public) Address is the public address of the firewall itself. The NAT Public address refers to the fact that the SonicWALL's IP address is used for services forwarded to the internal network in a NAT environment. For example, if you forward PPTP to an internal Windows 2000 RRAS server using the Access Page, users would configure the SonicWALL's WAN IP address as the address of the RRAS server and rely upon it to forward the PPTP streams to the interior server.

The General Time page, shown next, allows you to configure your time zone, enable automatic time updates via the Network Time Protocol, and set the time manually.



The General Password page allows you to change the admin password for the firewall. The standard built-in management account is called admin and it has the password you set previously, when you configured the firewall using the Configuration Wizard.

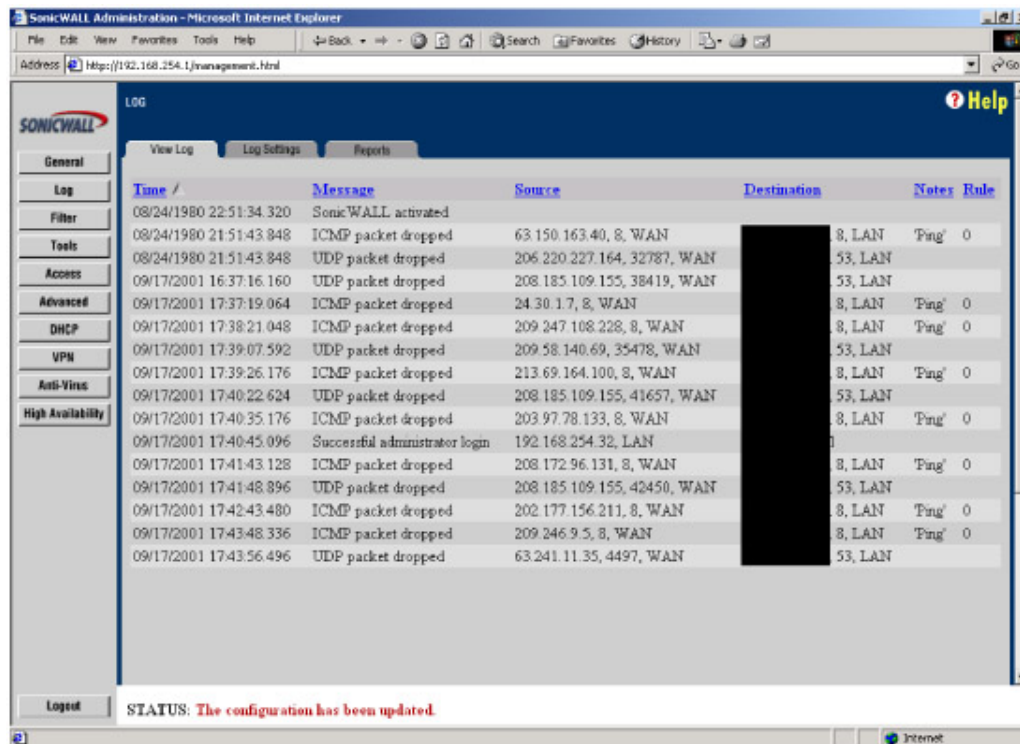


Log

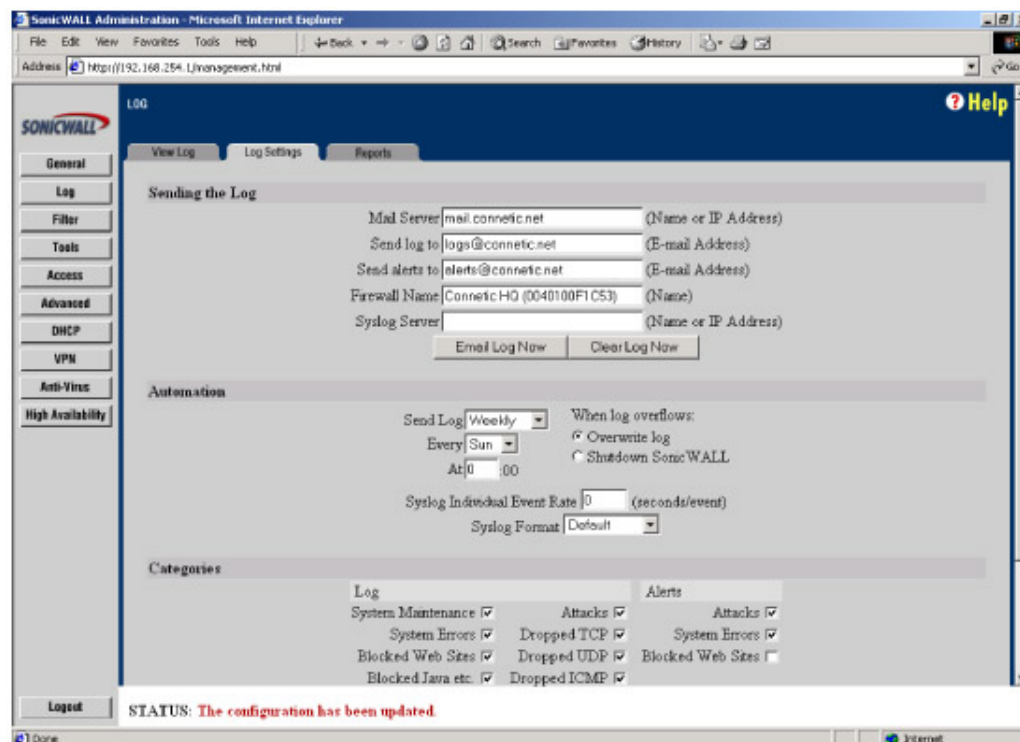
The Log section provides access to and settings for the SonicWALL firewall log. SonicWALL firewalls provide a very complete logging mechanism, including automatic interpretation of numerous types of attacks. Unfortunately, as with all automatic analysis systems, the firewall is frequently incorrect about the types of attacks that may have occurred, so you should rely on its

analysis only as a general guideline.

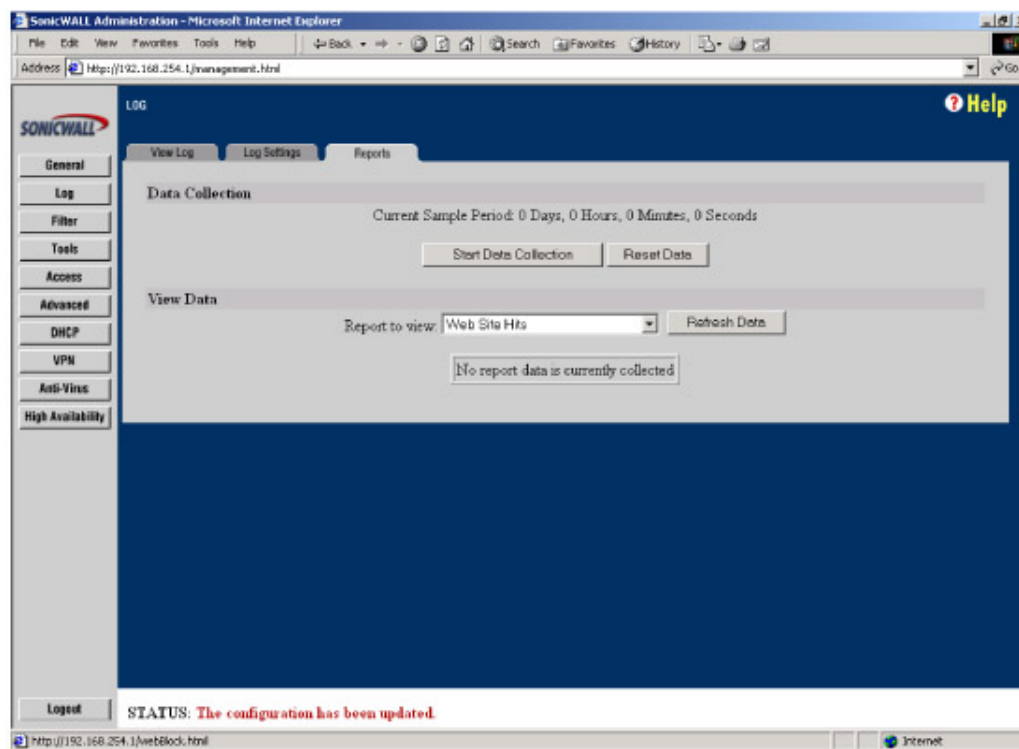
Here is the Log View page, which presents the entire firewall log for your perusal.



The Log Settings page, shown next, allows you to set up log forwarding to an e-mail account and to configure event forwarding to a Unix syslog server. The CD-ROM that accompanies the firewall has a syslog daemon for Windows as well. You should strongly consider forwarding all your firewall logs to a central account or to the same syslog server for centralized management.

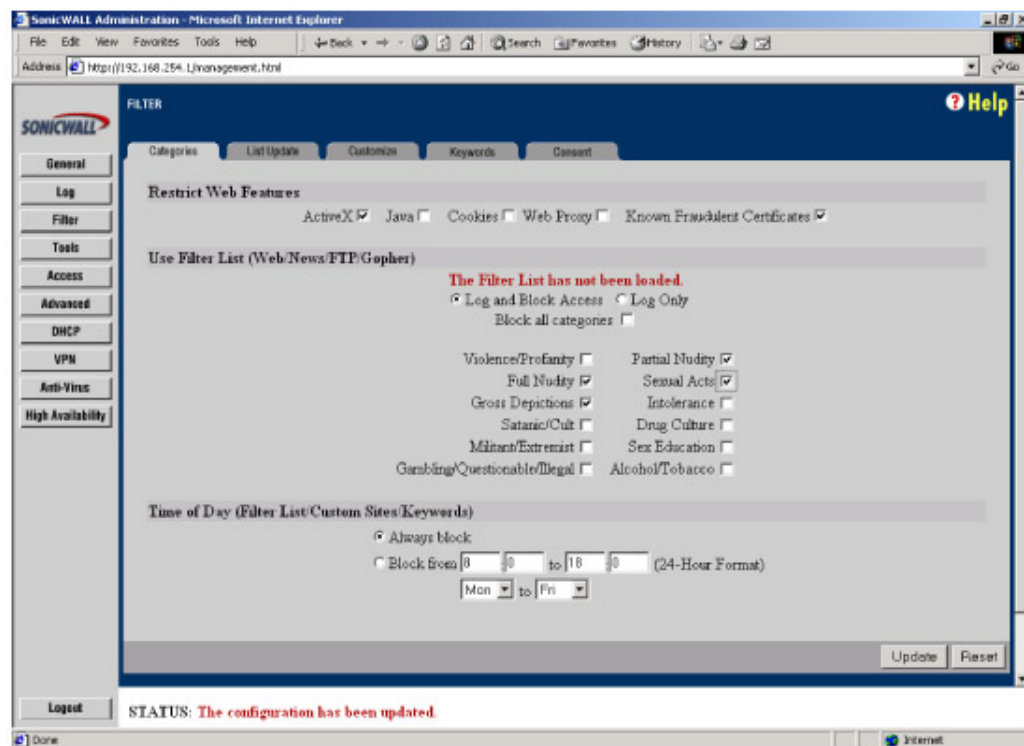


SonicWALL devices also provide rudimentary support for reporting. In the Log Reports page you can select a preconfigured report and start collecting. Once collection has been started, the device will build a report until you stop it. The report will then be available for viewing on this page.



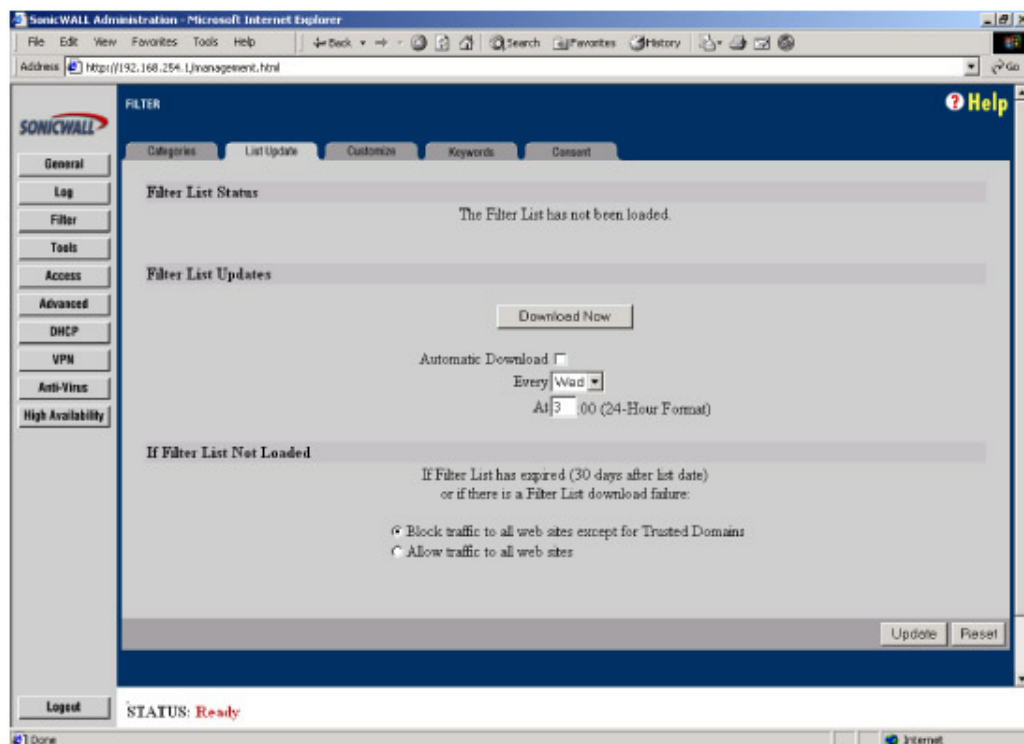
Filters

The filter section allows you to strip ActiveX controls, Java applets, cookies, and a few other miscellaneous content items out of HTTP streams. This is all the filtering that a SonicWALL can perform out of the box. The next screen shows the web restriction features. Other protocols and more sophisticated HTTP filtering are not supported, but you can redirect HTTP to a proxy using the Advanced Proxy Page.



These pages are primarily geared toward those customers who have subscribed to SonicWALL's filter service to configure website blocking and reporting service. This screen shows the categories of websites that a SonicWALL can filter if you subscribe to the web filtering service.

The next screen shows the Filter List Update page, which allows you to customize how the blocked sites filter list is downloaded and what action to perform if the filter list expires.

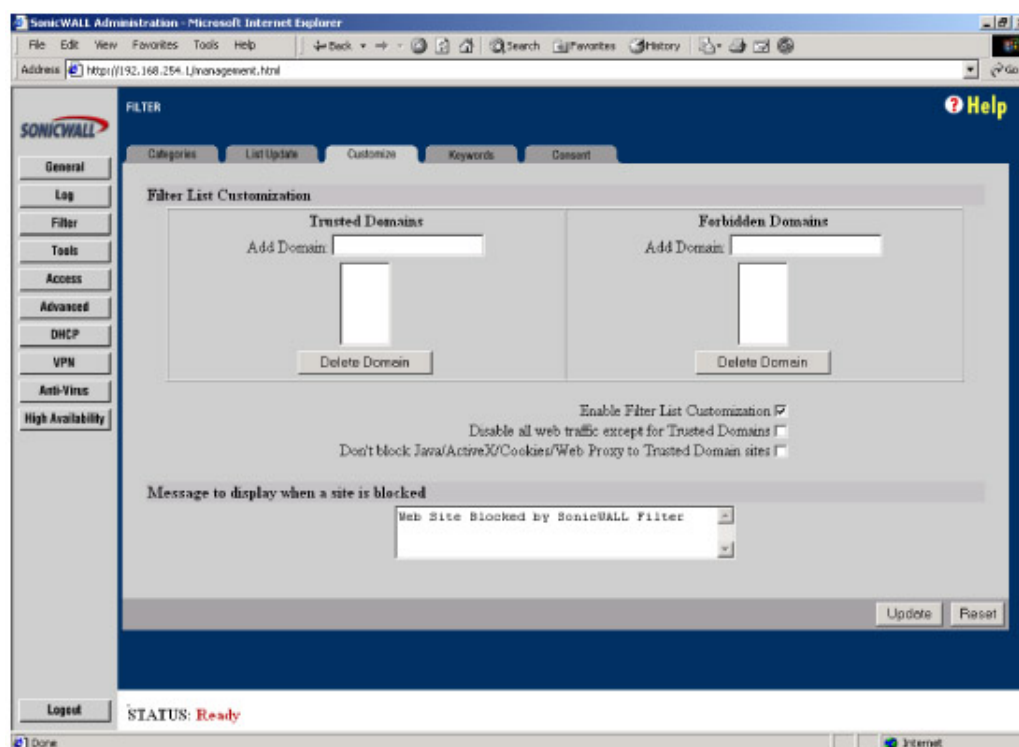


Frankly, site filtering based on a list of "banned sites" is waste of time in my experience. If someone inside your organization wants to get access to the type of information you're blocking, they need only use a search engine to find sites that aren't contained in your filter list. Site banning actually

winds up being a silly escalation, with users feeling that they're "allowed" to visit any site that isn't explicitly banned.

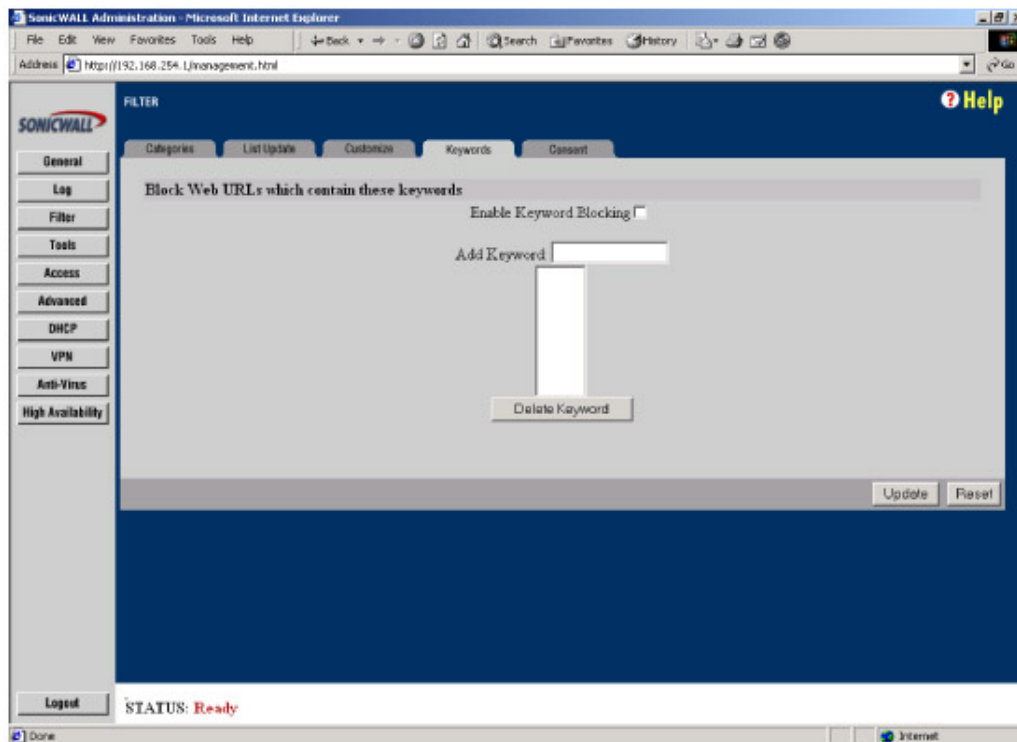
Rather than banning sites, inform users (using the Filters Consent page) that the IT department monitors all sites visited and that they will be held accountable for their Internet usage. This prevents problems before they become problem and encourages people to restrict their own activities. You can monitor which sites are visited using the Log Reports page.

The next screen shows the Filter Customize page, which allows you to specify trusted domains that users should always have access to, and to specify additional sites to block that aren't contained in the downloaded blocked sites list.

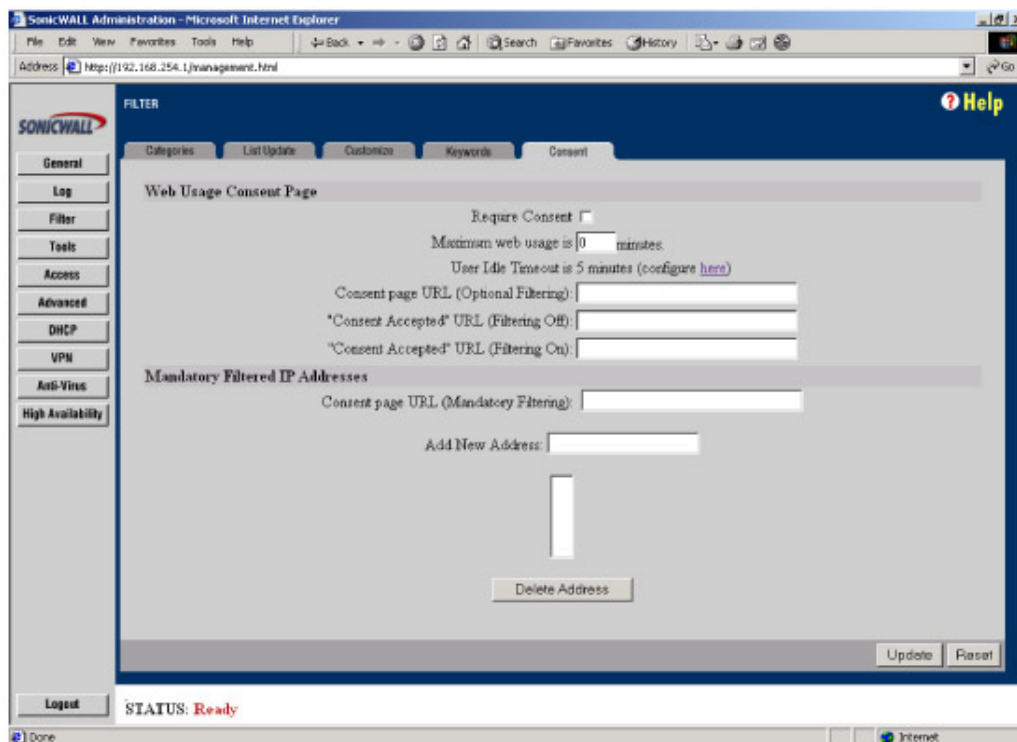


What isn't obvious about the Filter Customize page at first glance is that you can configure the firewall to block every site except for a few trusted domains without paying for the filter list. By setting the firewall to block sites when the list is expired, then adding sites you wish to allow to the trusted domains list, you can ensure that users will only be able to visit sites you allow. For organizations where "Internet" access isn't appropriate but "extranet" access is, this can create an effective access list.

Keyword blocking is far more effective in restricting websites than site banning in general, but SonicWALL's keyword filtering applies only to the contents of the URL, not the contents of the page. You'll need a proxy server that supports keyword blocking if you want to use true keyword blocking in your organization.



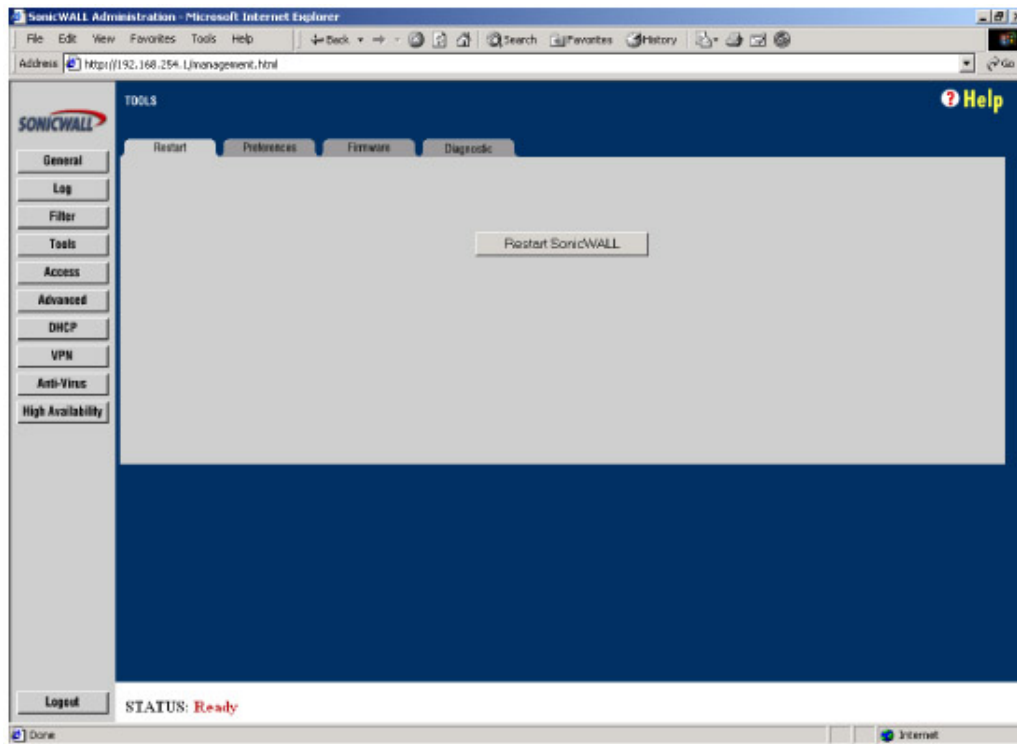
SonicWALL firewalls support enforcement of per-session consent before users can browse the Web. This allows you to put your company's Internet usage policy in front of each user every time they browse the Web and force them to click an Accept button indicating that they understand the policy before they proceed. This disallows legal defenses that claim a lack of knowledge or understanding on the part of employees should a situation ever progress to that point, and more importantly reminds people of their responsibilities before they browse.



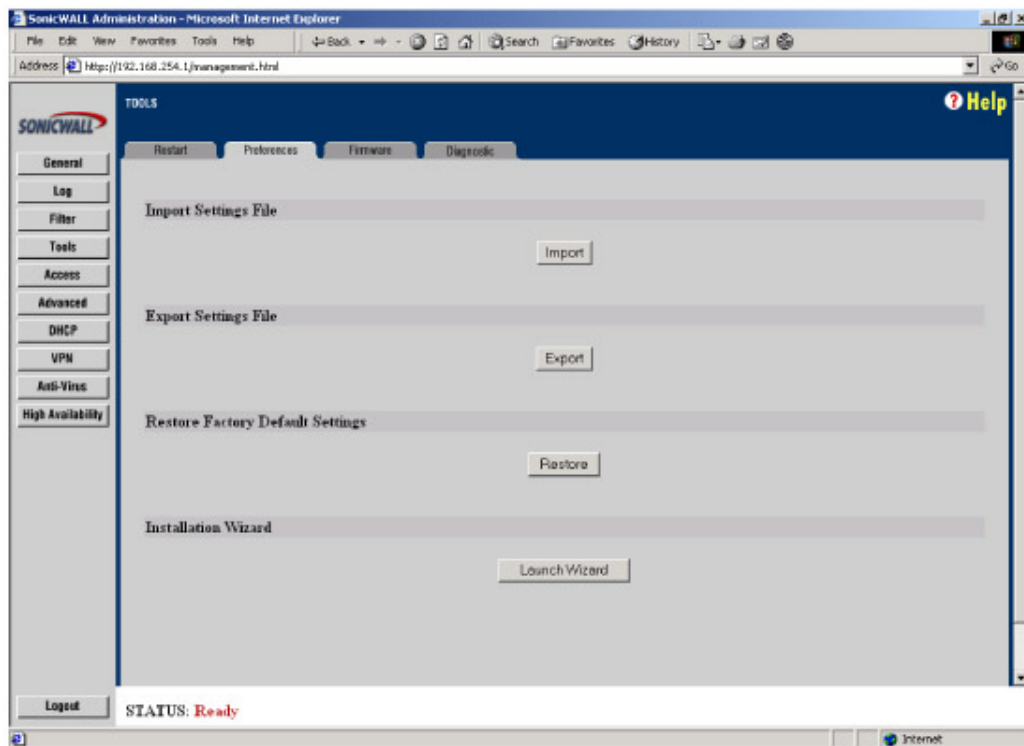
Tools

The Tools area allows you to perform many utility functions that are common on general-purpose computers, like pinging sites, looking up names, and performing trace routes, as well as saving and importing settings, restarting the firewall, etc.

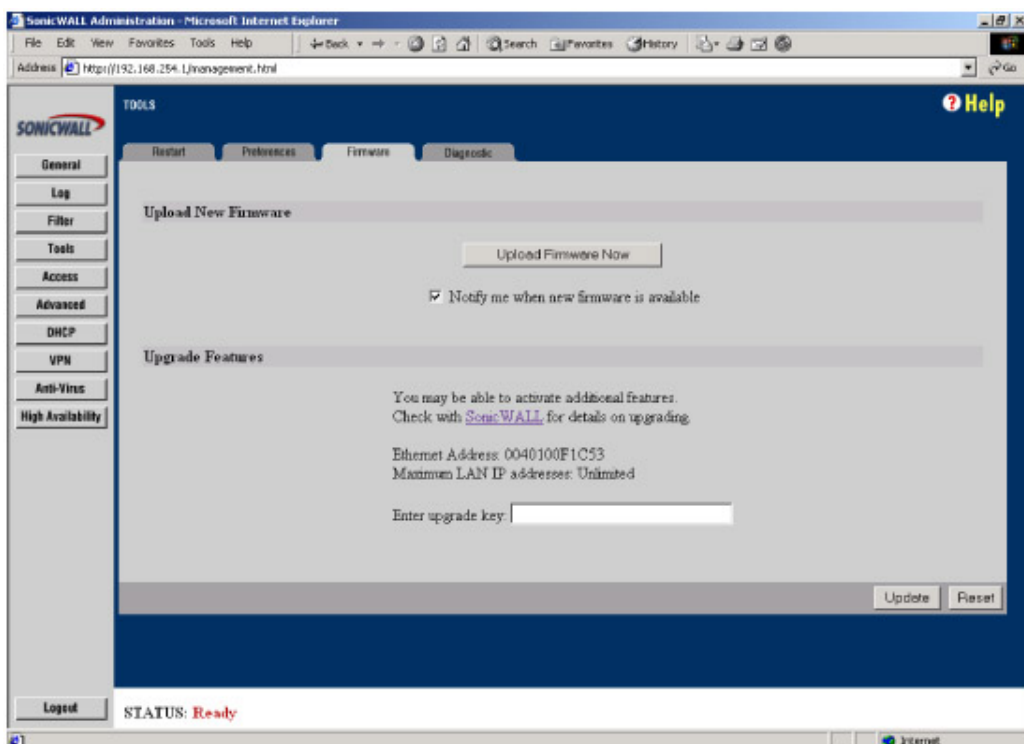
This screen shows the Tools Restart page, which as you may suspect allows you to reboot the firewall. You'll need to do this after performing certain updates.



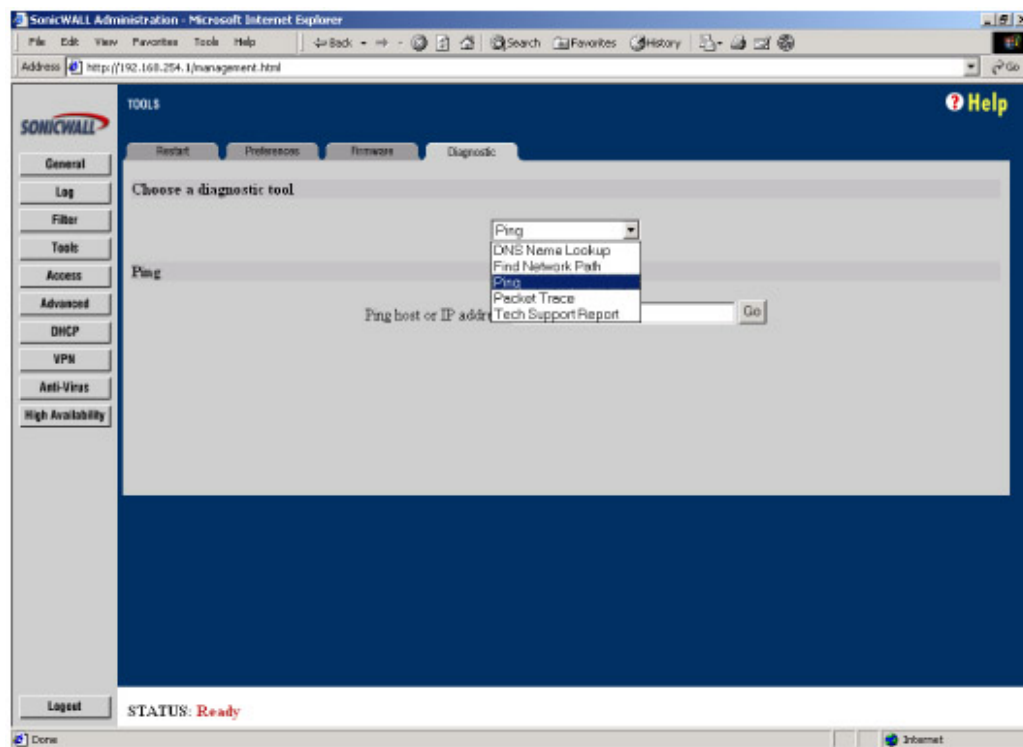
The Tools Preferences page allows you to import and export your settings for the purpose of backup. You can also clear the firewall settings (except the password and LAN IP address) and relaunch the Configuration Wizard if necessary.



The Tools Firmware page, shown next, allows you to install a downloaded firmware update. For the firewalls, SonicWALL typically releases new firmware about twice a year. Each update typically includes a few minor (and some major) functional enhancements, and some bug fixes, and otherwise incrementally improves the device. In the four years that I've used SonicWALL devices, I've seen their reliability and feature set improve noticeably due to firmware updates. To install an update, go to the SonicWALL website and download the latest update for your device. Then click the Upload Firmware button to install it from your administrative computer. It would be handy if the firewall could actually download the firmware directly over the Net, but that would not be as safe, since an interruption in the download would leave the device in an unusable state.



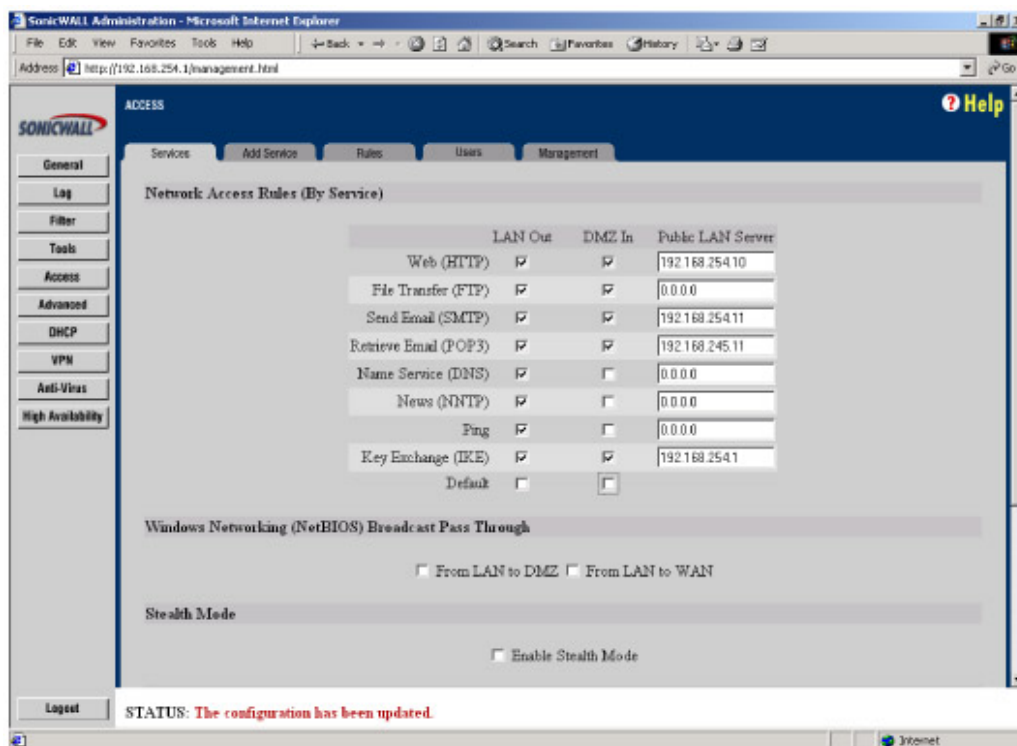
The Tools Diagnostics page, shown next, provides the typical set of Unix TCP/IP diagnostic tools. To use a tool, select it in the pick box and the page will change to accept the appropriate type of input.



Access

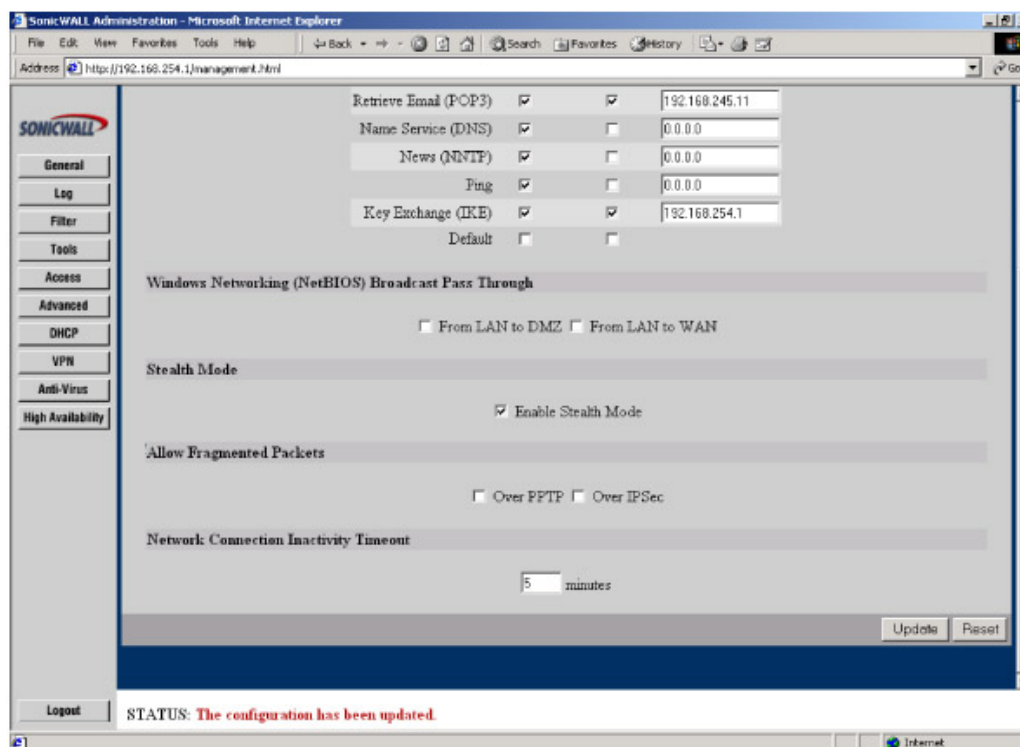
The Access pages are where the bulk of the firewall configuration occurs. SonicWALL firewalls use the same rule-based firewall management techniques that stateful inspection firewalls have used since the beginning of the Internet Age, but they've improved the ease-of-use by putting a simple service-enable and port-forward paradigm in front that automatically generates the rules for you.

The Access Services page allows you to allow specific services into either the LAN or DMZ and to forward ports on the Firewall's WAN IP address to hosts inside the NAT addressed LAN. It's not entirely clear by looking at this page, but if you enter a host address into the Public LAN Server field, the port will be forwarded to the interior host you specify. To enable LAN Out access, check the LAN column for the service, and to allow the service into the DMZ, check the DMZ column for the service.



Unfortunately, it's not possible with this interface to allow only specific protocols to specific machines in the DMZ, but you can craft custom rules to perform this in the Access Rules page (covered later in this section).

The next graphic shows more Access options configurable on the Access Services page. NetBIOS Broadcast passthrough is useful for resolving Windows name-resolution problems between the LAN and the DMZ or WAN—however, it can allow dangerous information to leak to hackers and consume bandwidth unnecessarily. In my opinion, it's safer to use DNS or WINS rather than broadcast name resolution, or to simply refer to extranet servers by IP address.

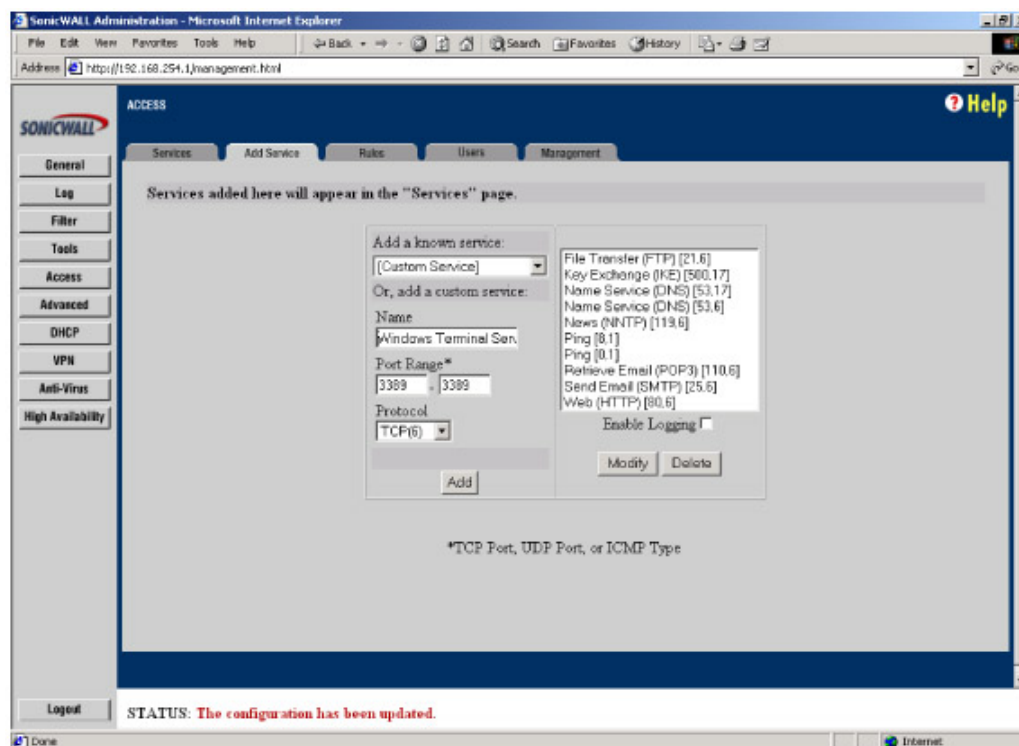


You can also enable Stealth Mode scanning, where the firewall simply fails to respond to ICMP requests. This keeps scanners from determining that anything exists on the IP address, and makes port and address scans take considerably longer if the hacker persists anyway. However, if you are forwarding ports to the LAN that will be revealed by a persistent port scanner, Stealth Mode will make it quite clear to a hacker that you are using a firewall. If you're going to set up a honeypot server inside your network, don't use Stealth Mode.

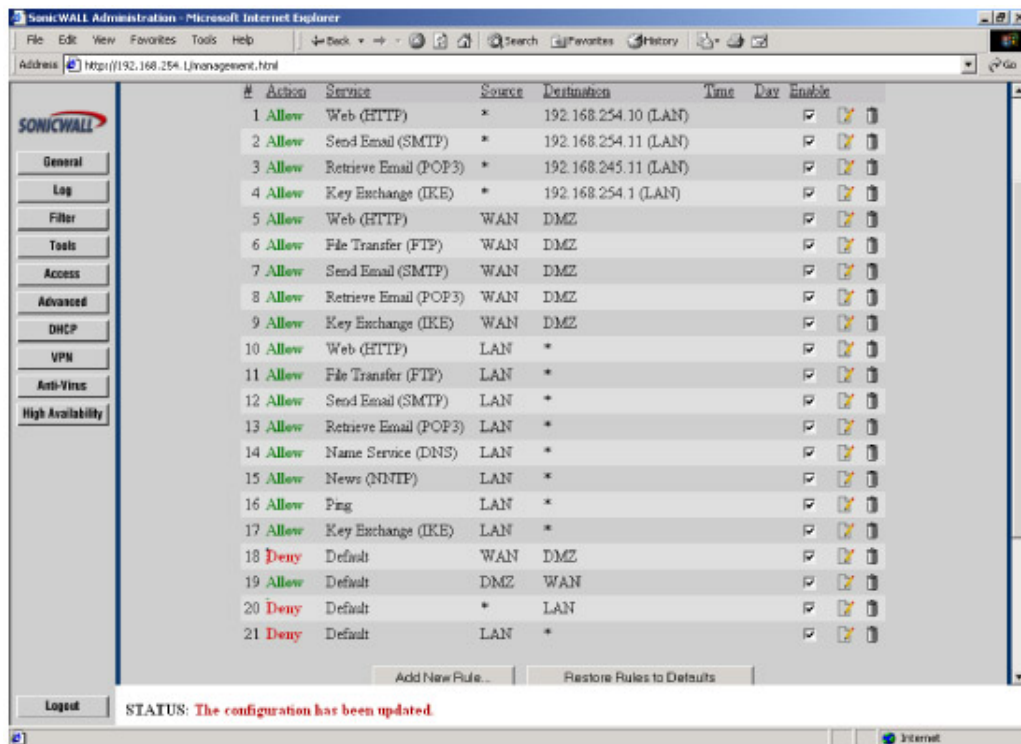
Fragmented packet handling allows the firewall to break up large packets for transit over congested WAN links. This can improve the reliability of a VPN connection at the expense of bandwidth.

The Network Inactivity Timeout setting specifies how long the firewall will maintain connections when they are unused. Longer connection times allow users to maintain web sessions, Telnet sessions, and other session-based connections with longer inactivity periods before the firewall drops the translation rules in its tables. Shorter periods may be required for busy firewalls, in order to make better use of the firewall's limited RAM. I recommend 30 minutes as a reasonable minimum to avoid dropping connections for users who have simply turned away from their consoles to work on something else for a few minutes.

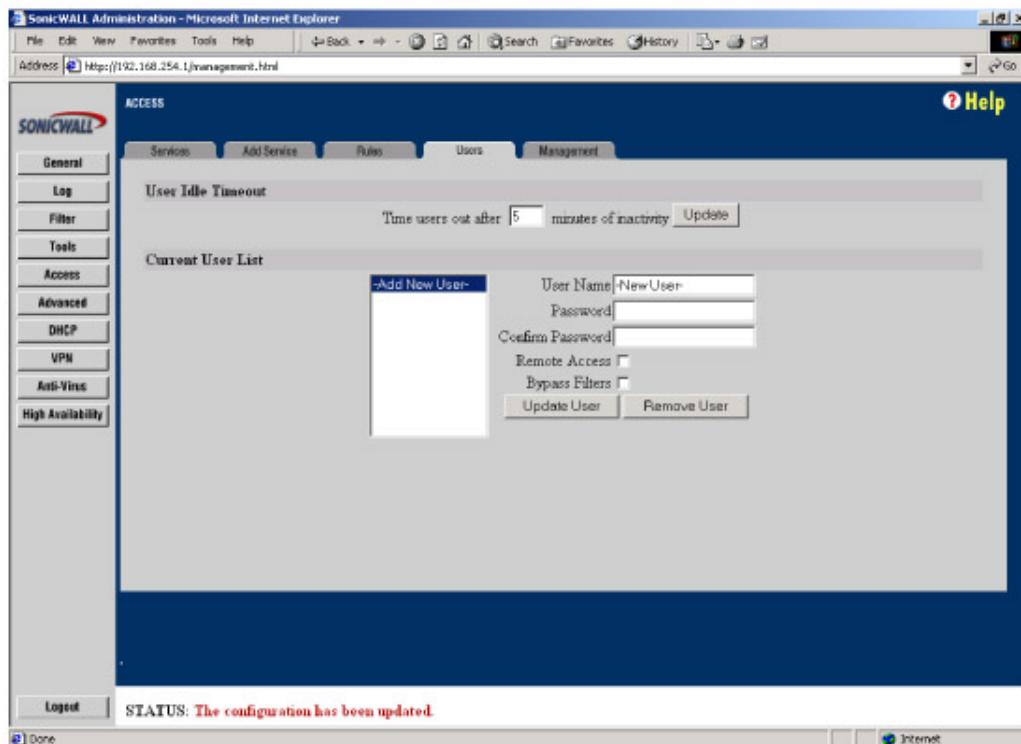
The Access Add Service page is used to add existing service definitions to the Access Services page or create a new service definition that can subsequently be controlled using the Access Services page. You can also control whether a specific service is logged or not with this page, which is shown here.



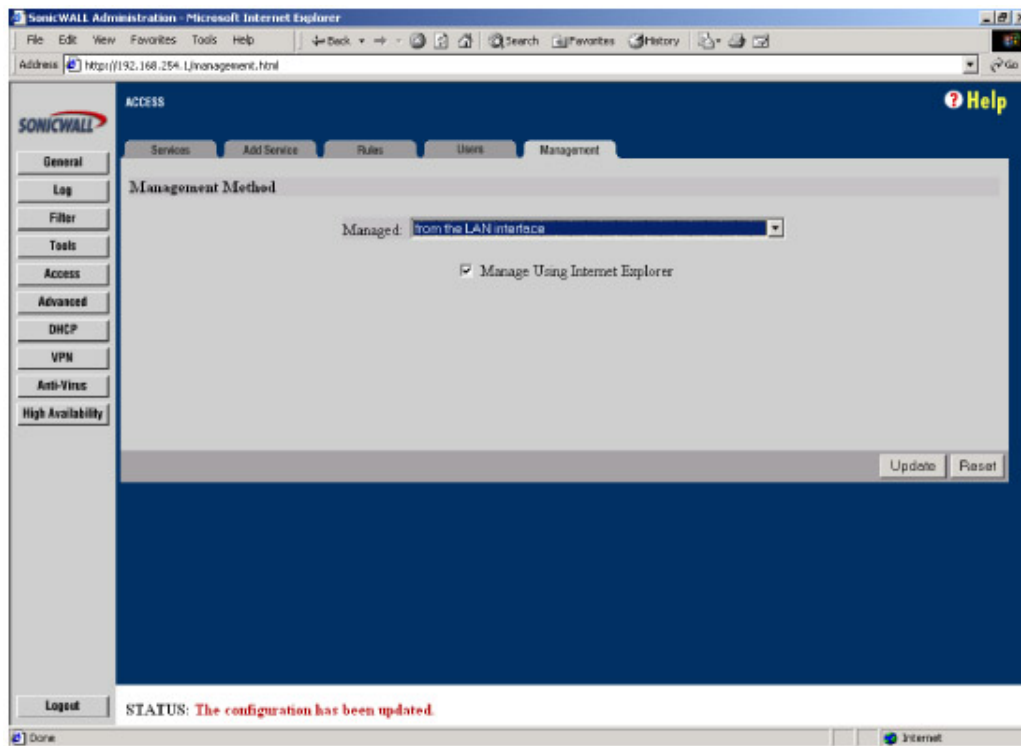
The next screen shows the Access Rules page, which is a typical rule-based policy editor similar to the ones used on most firewalls. Unlike most firewalls, you'll rarely need to use the Rules Policy page for SonicWALL, because nearly all access requirements can be more easily defined using the Access Services page. Use the Access Rules page when you need to define custom rules beyond those that can be controlled by the Access Rules page, such as controlling specific services to various machines within the DMZ.



The next graphic shows the Access Users page that you can use to create login accounts for the firewall. You'll need to create user accounts if it's important to have multiple people manage the firewall without revealing the admin password. However, as with all account management systems, it's crucial not to allow the number of users and accounts to grow beyond a manageable number or to leave obsolete accounts active in the system. For most medium-to-small sites, there's little need to use multiple management accounts.



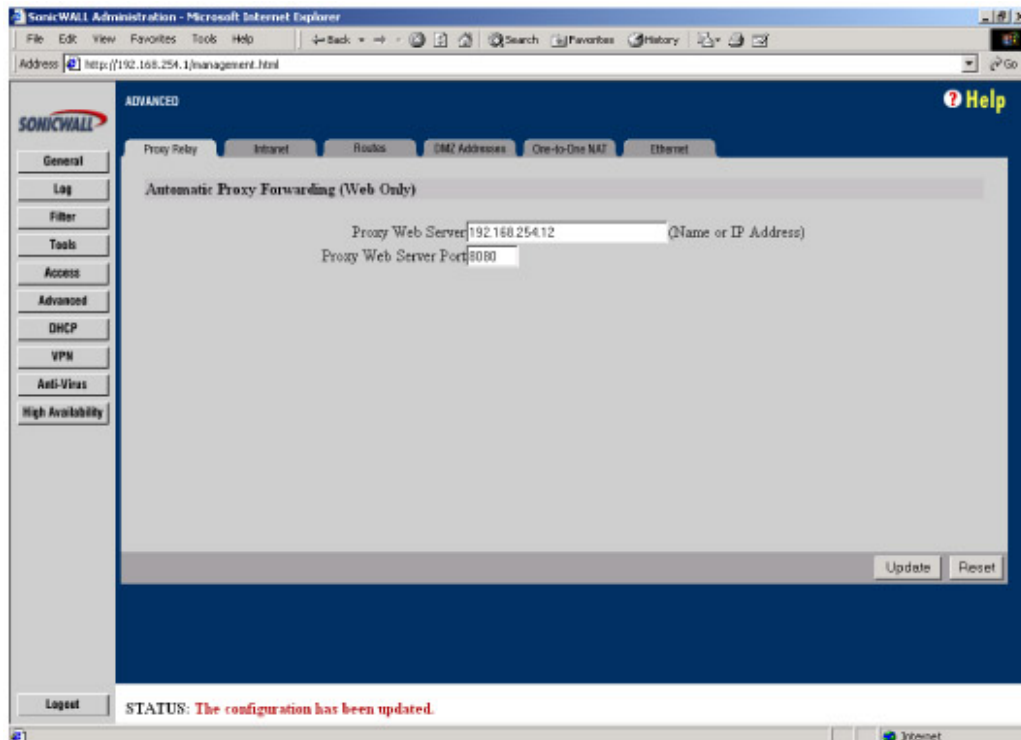
The next screen shows the Access Management page, which is used to determine how the firewall can be managed. There's also a check box to indicate if you use Internet Explorer to manage the firewall, which will improve the download speed of the login Java applet.



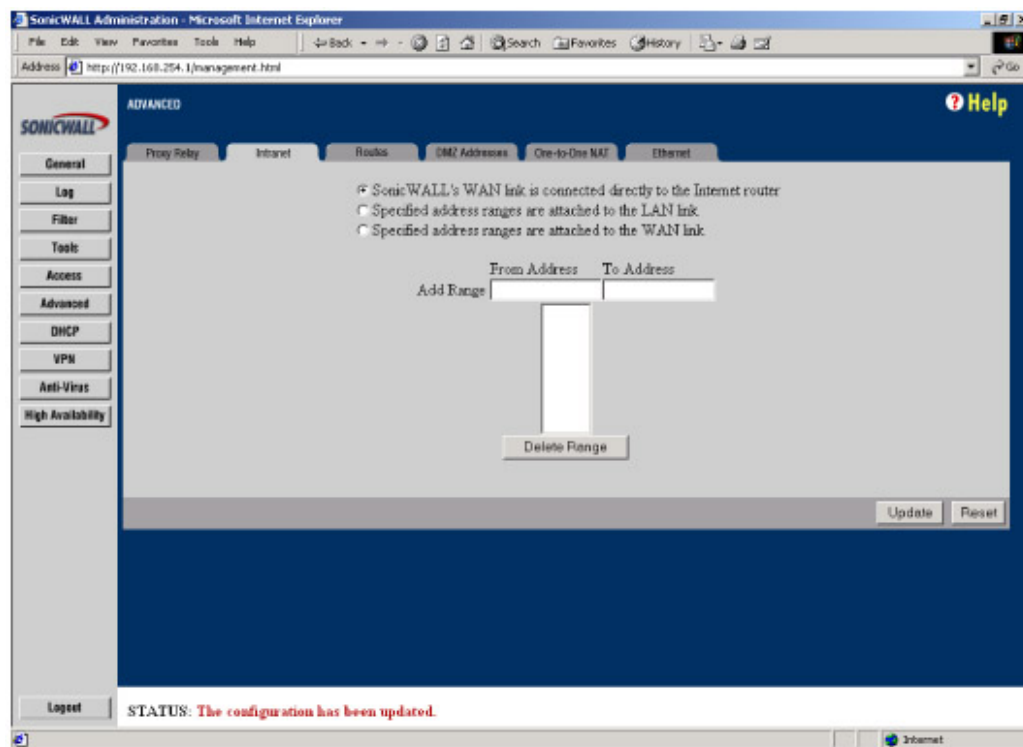
Advanced

The Advanced topical area is used to configure the firewall beyond firewall access policy and basic network configuration.

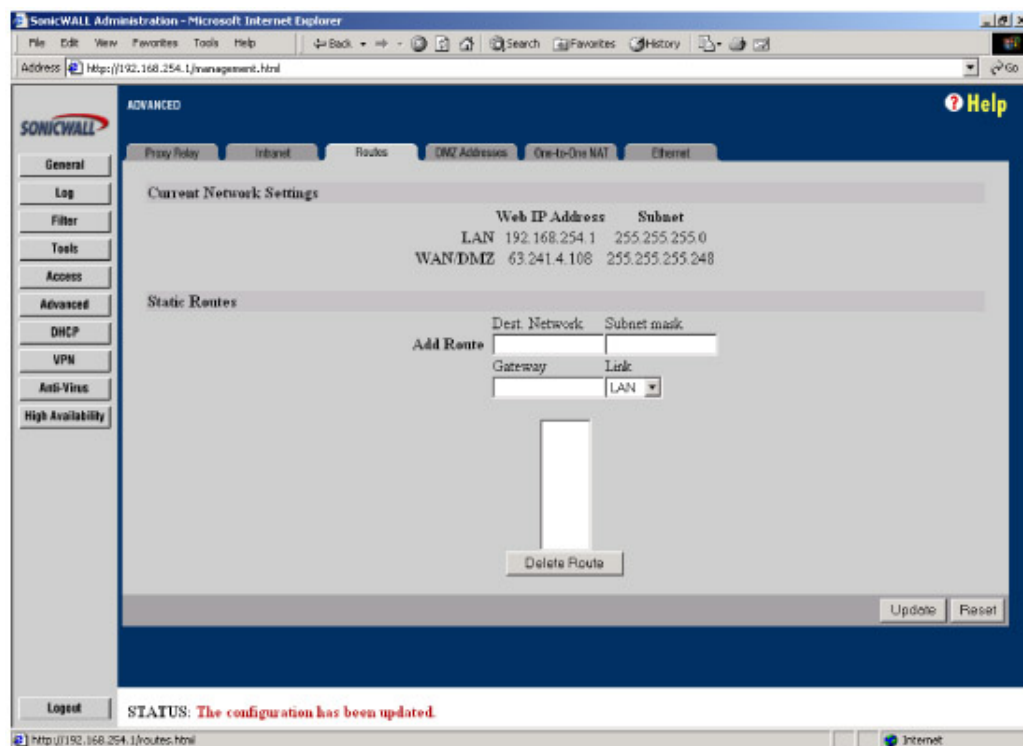
The Advanced Proxy Relay page lets you specify a proxy server through which all HTTP connections must be routed. This allows you to shore up the firewall's deficiency in filtering HTTP streams by forwarding them to a more sophisticated HTTP relay.



The Advanced Intranet page is used to define the interior routing structure of the network. This allows the firewall to operate properly with a network architecture that's more sophisticated than a single IP subnet.

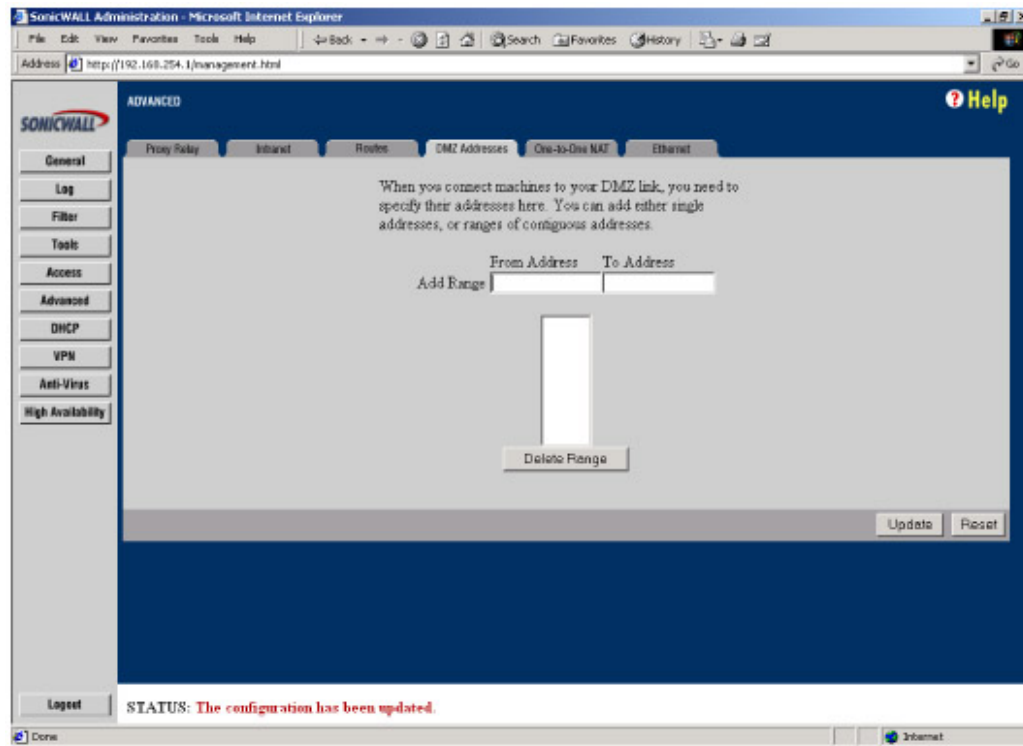


For even more complex routing architectures, the Advanced Routes page, shown here, can be used to specify routing. The Advanced Routes page is analogous to using the route command on Windows or Unix hosts.

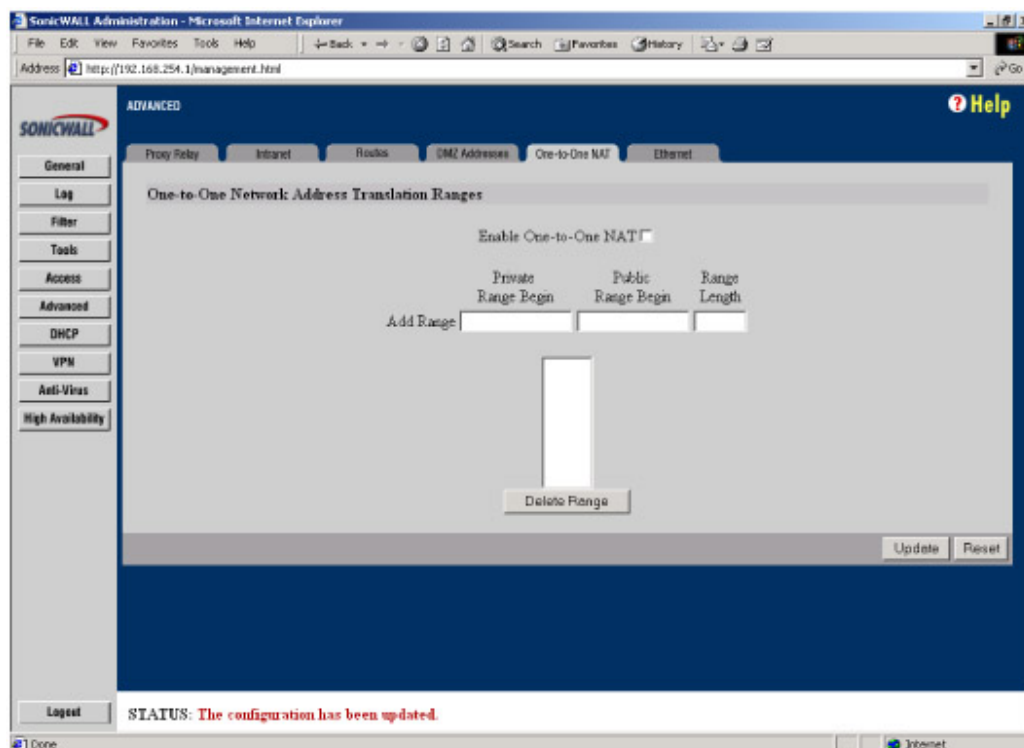


The Advanced DMZ Addresses page is used to define those address ranges you want to be filtered through the DMZ port. For example, if your ISP assigns you a /28 network with 13 available

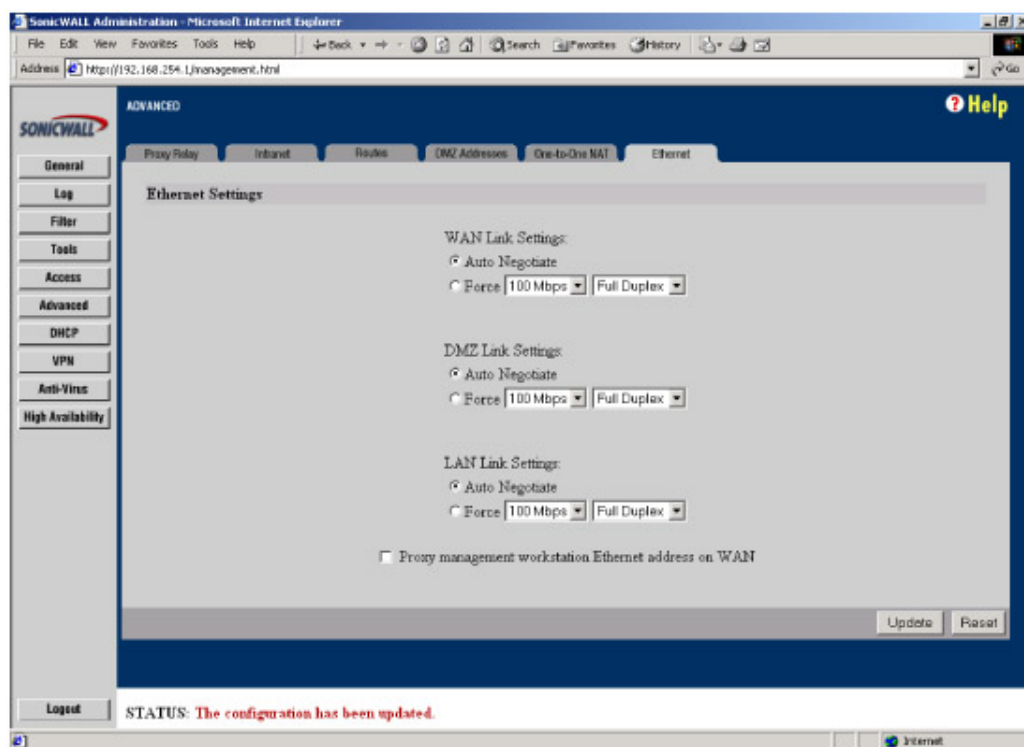
addresses, you would probably assign the firewall to the .2 address (assuming the default gateway is on the .1 address) and put the .3–.14 addresses in the DMZ range in this page so that servers on those IP addresses would be correctly filtered.



The next screen shows the Advanced One-to-One NAT page. 1:1 As we've covered earlier, NAT is used to translate a block of IP addresses to another set of addresses of the same size. This is useful in cases where you have a large block of public IP addresses that must be translated to a sequential block of reserved addresses or in the case where your organization standardized on a set of IP addresses that have been permanently assigned to another organization and it would be inordinately difficult to change them. Using 1:1 NAT, you can simply translate them at the firewall.



The Advanced Ethernet page, shown next, shows port connection status for each of the firewall pages. Although the Auto Negotiate setting is nearly always appropriate, there are some older Ethernet devices that do not negotiate properly. In this case, setting the firewall's ports to specific settings will force the other device to act correctly.



DHCP

SonicWall Firewalls can be configured to serve DHCP addresses to clients in the LAN. I prefer to use firewalls as DHCP servers because they're far more reliable than general-purpose servers.

Here is the DHCP Setup page.

SonicWALL Administration - Microsoft Internet Explorer

Address: <http://192.168.254.1/management.html>

SONICWALL DHCP Help

Setup Status

General

☒ Enable DHCP Server

Lease Time: 60 minutes

Client Default Gateway: 192.168.254.1

Subnet Mask: 255.255.255.0

DNS

Domain Name: hq.conectic.net

☐ Set DNS Servers using SonicWALL's Network settings, or

☒ Specify manually

DNS Server 1: 240.3.32

DNS Server 2: 240.3.33

DNS Server 3: 240.3.34

WINS

WINS Server 1: 0.0.0.0

WINS Server 2: 0.0.0.0

Dynamic Ranges

192.168.254.32 - 192.168.254.254

Range Start:

Range End:

Logout

STATUS: The configuration has been updated.

You can determine which MAC addresses are assigned to which IP addresses using the DHCP Status page shown next. This is also useful for de-referencing a MAC address from an IP address when you're performing network troubleshooting.

SonicWALL Administration - Microsoft Internet Explorer

Address: <http://192.168.254.1/management.html>

SONICWALL DHCP Help

Setup Status

DHCP Leases

Current: 1

Available Dynamic: 222

Available Static: 0

Total: 223

Current DHCP Leases

192.168.254.32 08:00:46:0D:A4:BC DYNAMIC

Delete

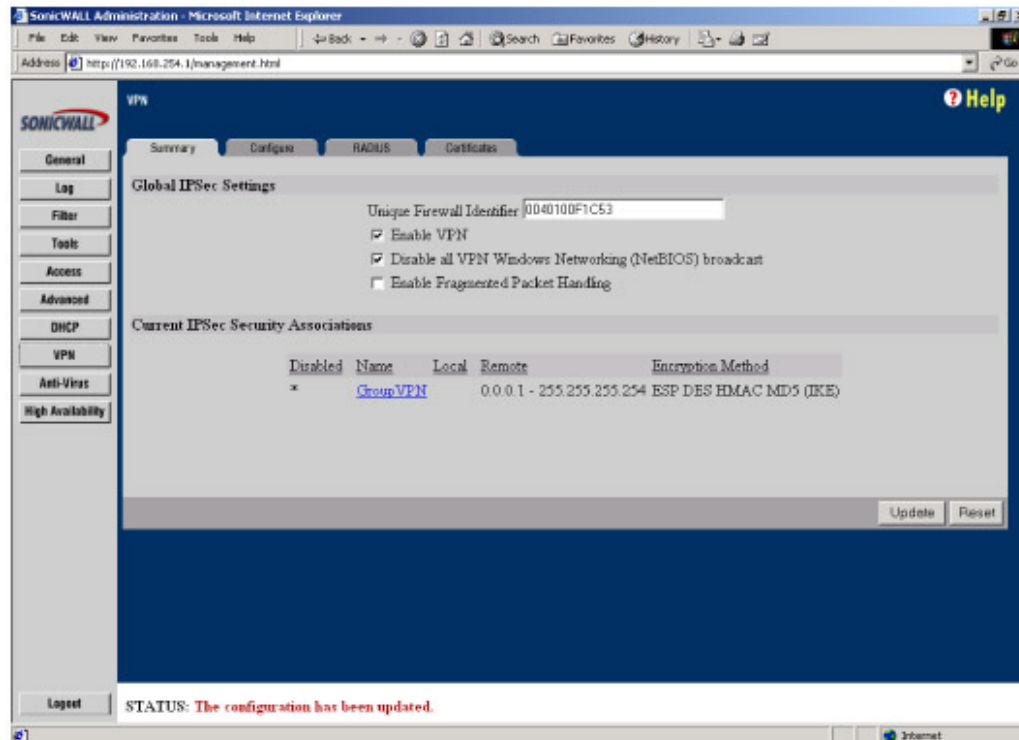
Logout

STATUS: The configuration has been updated.

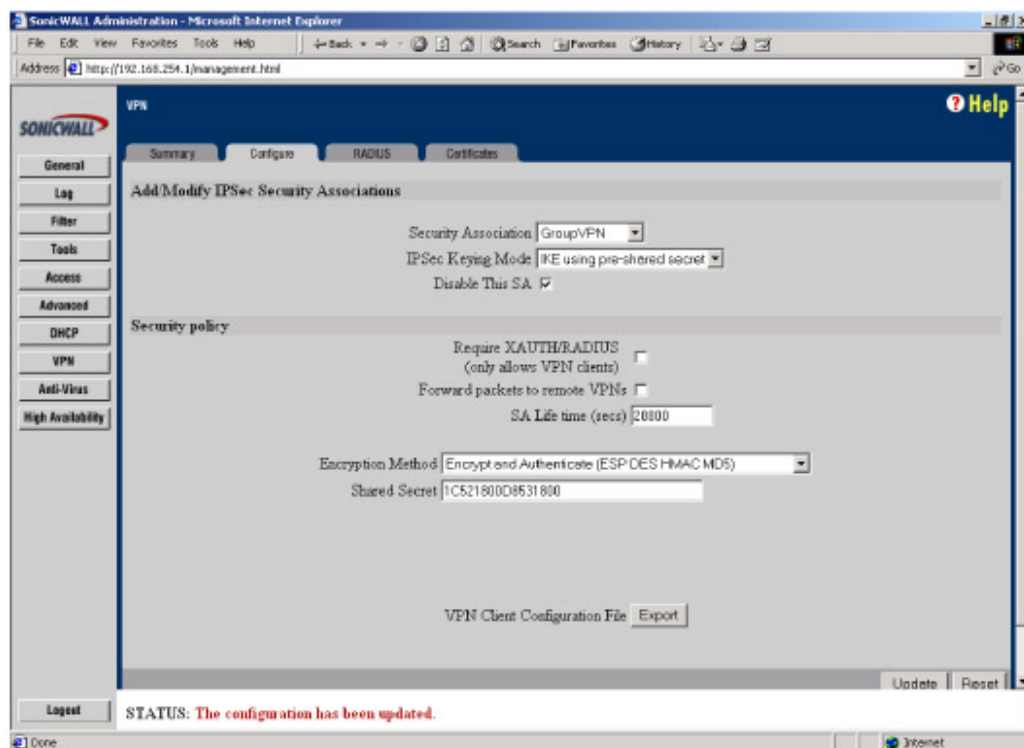
VPN

Firewalls make outstanding platforms for providing Virtual Private Networking services—in fact, they're the only good place to do it, because separating the firewall and VPN functions would mean that your VPN weren't firewalled. This condition could subject the VPN devices to information leaks and denial-of-service attacks that would be automatically prevented in a unified device. Using firewalls for VPN platforms also simplifies routing configurations and eliminates the (strong) possibility of conflicts between the firewall and VPN device that can occur when both devices make assumptions about how traffic should be handled.

The VPN Summary page shows general VPN settings and a status line for each defined Security Association (SA).

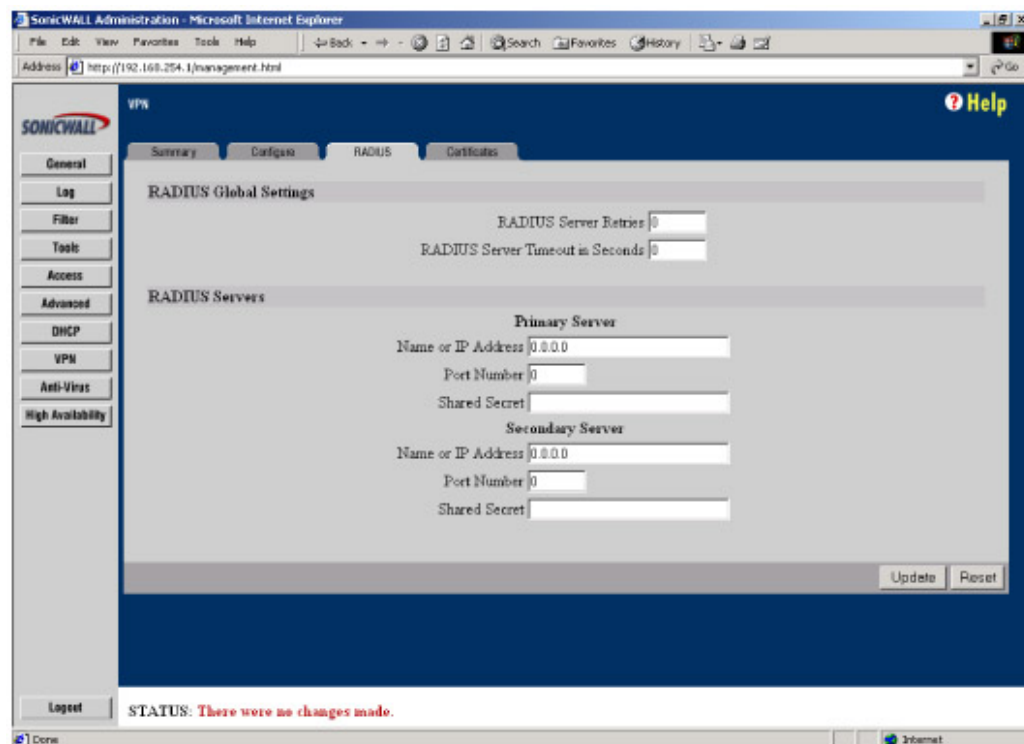


You can create Security Associations using the VPN Configure page shown in the next figure. Defining an SA is relatively simple: you simply select a mode (IKE or manual key), enter the address of the remote VPN gateway, and enter the keying material—either a secret key for IKE or a public key for IPSec manually keyed VPNs.

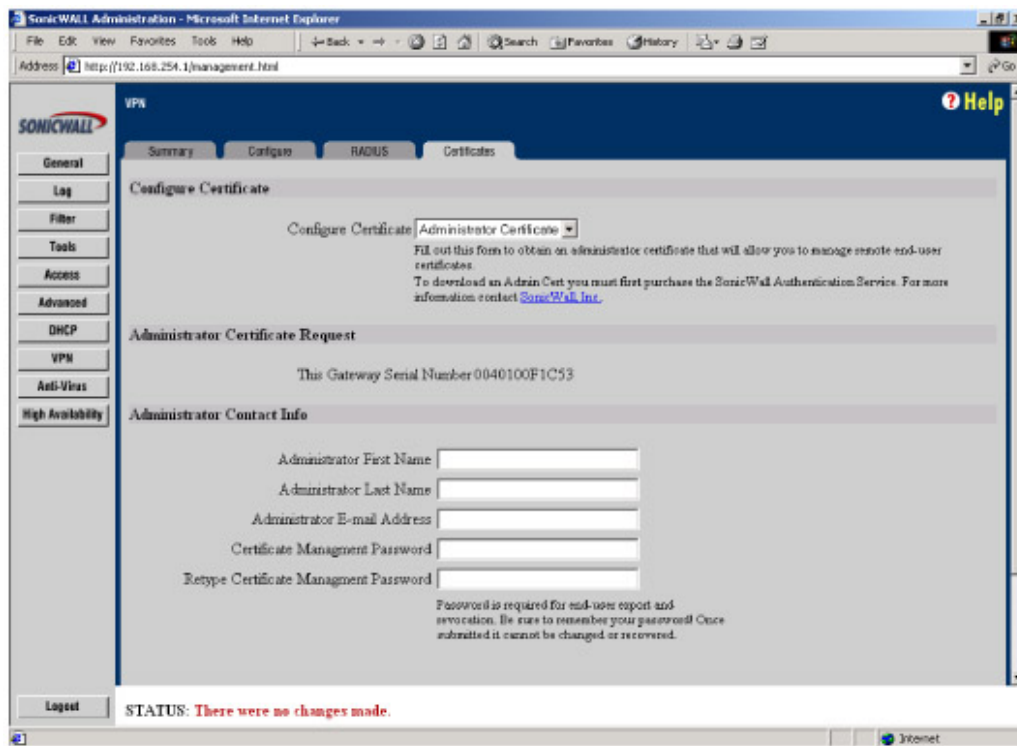


The VPN Configure page is somewhat quirky. Frequently, the selected SA will change unexpectedly, so if you're not paying attention you might wind up incorrectly configuring both the SA you intended to configure and the SA that the firewall switched to.

The next figure shows the VPN RADIUS Configure page. If you want to use RADIUS to authenticate remote access users, you can configure the address of the RADIUS servers with this page.



SonicWALL firewalls can use certificates for remote access authentication in addition to IKE with secret keys or manually keyed IPSec. However, you must subscribe to SonicWALL's certificate service in order to generate the required certificates. Here is the VPN Certificates page.



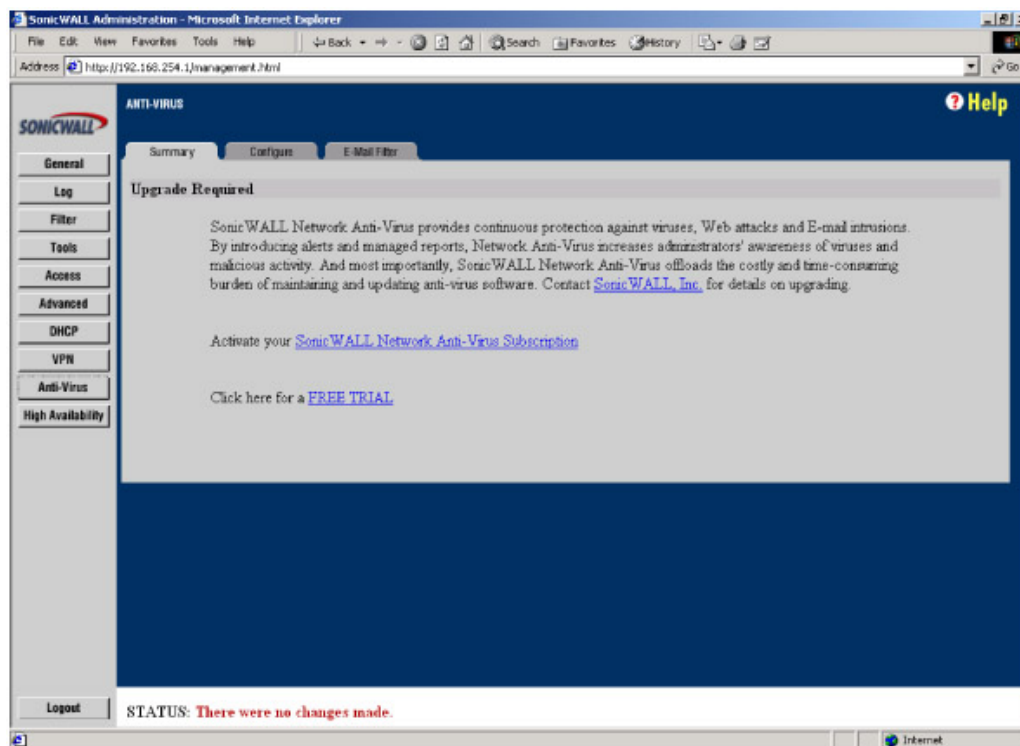
Anti-Virus

SonicWALL Firewalls provide the option of subscribing to a virus detection service that operates on the firewall and filters those inbound connections that contain files.

While convenient, device-based anti-virus scanning is not a substitute for desktop-based virus scanning. The problem is that there are more vectors for virus infection than just the Internet. Viruses can come in from removable media like floppies or CDs or can walk in on portable laptops that connect to the Internet at other sites. Finally, a virus contained in a compressed file will not be detected even if it is downloaded through an anti-virus filtered firewall.

Desktop-based anti-virus scanning solves all of the above problems, and should be used whenever virus intrusion is a problem.

If you subscribe to SonicWALL's Anti-Virus Summary page, you'll configure it using this topical section via the Anti-Virus Summary page.

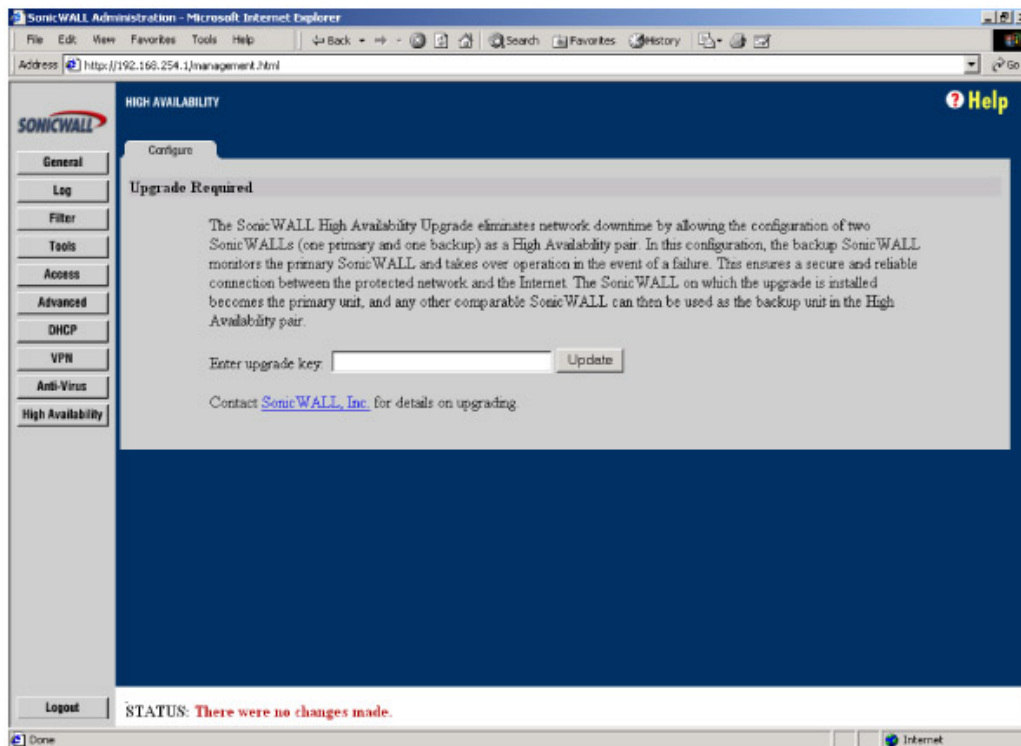


High Availability

SonicWALL firewalls support a high-availability feature that allows you to place two identical SonicWALL firewalls in parallel. In the event that one firewall fails or locks up, the other firewall will immediately take over to maintain Internet connectivity.

In the years that we've been supporting SonicWALL firewalls, we've seen them cause an average of about an hour of Internet downtime per year, which means that they're about 99.99% reliable. This downtime is usually due to an external circuit failure that causes an internal crash that can be solved by simply restarting the firewall manually. The one failure that took more time to solve required opening the device, performing a procedure to erase the flash memory, uploading new firmware, and re-importing the backed-up firewall policy. To this day, I have no idea what happened to cause the problem, but it was two full firmware versions ago, and I haven't seen the problem since.

My point is that the high-availability option is only really necessary in organizations that can't tolerate a few hours of downtime per year. Since you'll probably never need to see it in real life, here is a screenshot of the High Availability Page.



So that's all folks! If you've never configured a firewall before, you can see now that the modern devices are neither confusing nor difficult to operate, and there's no mystical knowledge or lifetime of experience required. Once they're configured, they require no maintenance until your security policies change, and will usually operate indefinitely because the only failure component in them is a fan. Keep them cool and clean and they'll last for your entire career.

Case Study: Scaling the Firewall

When I first started using firewalls, they were just filter sets on routers that you telnetted into to configure. The sophisticated ones, like the Cisco PIX, worked the same way but had more features and stronger security. Then, when heavy-duty Internet hacking was invented, along came Application layer firewalls like TIS Gauntlet. It was a Unix behemoth that worked like every other Unix server application: You set cryptic configuration settings in cryptic files using cryptic text editors and controlled the operation of the firewall using cryptic command-line programs. Firewall-1, the first Windows-based firewall I used, was a breath of fresh air compared to the first generation of Unix-based firewalls. But it too had a bit of a learning curve: Why create "objects" to represent networks and computers when I could just as well use IP addresses? Why make abstractions called "services" for what were obviously sockets? Of course, lost to me at the time was that when you did this, actual humans could read and understand the rule base. Major new features, like Network Address Translation and VPN, were coming out every year at that time, forcing constant upgrades.

Then came the devices, with SonicWALL leading the way. These firewalls sported Firewall-1 style stateful inspection, but without all the installation hassle and without the expensive Sun or Dell computer. You just turned them on, and they worked! That was the idea anyway. I was worried about upgradeability for future threats, but the devices were so inexpensive I figured that, in the worst case, they could just be replaced.

The user interface was so simple and easy to use compared to previous offerings that any administrator with an understanding of IP could easily use it. The interface was a huge step forward in ease of use for firewalls, made even better by the fact that you didn't have to install a proprietary management client, as many of the high-end firewall offerings required. The device market has

come a long way since then, having all but destroyed the traditional firewall software market. Nearly all software vendors now either package their firewalls preconfigured along with computers and call them devices or have licensed their code to device manufacturers in order to compete—or have disappeared. But the SonicWALL is largely the same. Only incremental improvements and a few service offerings have been added since its inception. The original idea is unchanged after four years in the market, which indicates that with these devices, the firewall market is mature and won't be seeing any significant changes any time soon.

Part III: Additional Security Tools

Chapter List

Chapter 12: Attack Profiles

Chapter 13: Security Utilities

Chapter 14: Intrusion Detection

Part Overview

Featuring:

- How hackers attack networks
- Tools hackers use
- Small security tools you should know about
- How to perform strong monitoring with standard PCs and firewalls
- How to detect network intrusions
- Tools to detect network intrusions
- Tools to search for network vulnerabilities
- Tools to mislead hackers
- How to respond to an intrusion

Chapter 12: Attack Profiles

Overview

This book discusses a number of potential attacks without necessarily defining them. This chapter profiles all of the common attacks hackers use to localize, identify, and attack your systems. Given the information about how these attacks work, you can configure sophisticated firewall logging and alerting mechanisms to detect them. This chapter is broken down into the following broad categories:

- Denial-of-service attacks
- Exploitation attacks
- Information gathering attacks
- Disinformation attacks

These broad categories describe the purposes of nearly all hacking attacks.

Denial-of-Service Attacks

Denial-of-service attacks attempt to prevent you from providing a service by crashing or overwhelming your service computers. Denial-of-service attacks are the easiest hacks to attempt, so they are quite common. This section lists the most common of these attacks:

- Ping of Death
- Teardrop
- UDP floods
- SYN floods
- Land
- Smurf
- Fraggle
- E-mail bombs
- Malformed Messages

Ping of Death

The Ping of Death is the granddaddy of all denial-of-service attacks. It exploits the fact that many TCP/IP implementations trust that ICMP packets are correctly formed and perform too little error checking.

Warning

Attack Status: The original Ping of Death is obsolete; undiscovered effective malformations of the ICMP echo request may exist, however. Also, many other SMB, RPC, TCP, IP, or ICMP malformation attacks called "nukers" perform similar functions.

Profile

In order to test the throughput capabilities and packet size restrictions, ICMP packets can be created having anywhere up to 64KB. This allows you to determine the actual maximum packet size between you and a target system on the Internet. In the early days of the Internet, this functionality was important because many routers had maximum packet size restrictions. In large ping packets,

the payload portion of the packet is filled with meaningless data. The maximum payload size is 2^{16} (minus packet data overhead).

Ping-of-Death attacks are propagated by creating a malformed ICMP echo request packet in which the claimed packet size exceeds the maximum possible size. Because the payload size indicator is 16 bits, yielding a maximum possible packet size of 65,535 bytes (the actual limit is around 65,500 bytes due to packet header data overhead), packets that claim to be larger than 65,500 bytes can cause TCP/IP errors in the receiving system.

In a typical TCP/IP implementation, when a packet header is read, the information contained in the header is relied upon to create a buffer for the payload. When the claimed size of the packet header plus payload exceeds the 64KB maximum defined by the TCP/IP specification, the TCP/IP implementation may crash due to memory allocation errors.

Defense

All standard TCP/IP implementations have been hardened against oversize packets, and most firewalls automatically filter these attacks. Windows versions since Windows 98, Windows NT since Service Pack 3, Linux, Solaris, and the Mac OS are all immune to standard variations of the Ping of Death. Configuring firewalls to block ICMP and any unknown protocols will prevent this attack. For this reason, this attack in its current form is already obsolete and has no real future.

Teardrop

Teardrop attacks exploit a potential weakness in the fragment reassembly process of some TCP/IP implementations.

Warning

Attack Status: TCP/IP implementations can easily be hardened against Teardrop, but many operating systems are still vulnerable to it.

Profile

A teardrop attack exploits implementations that trust information in the packet headers of IP fragments. IP fragments contain fields that indicate which portions of the original packet the fragment contains. Some TCP/IP implementations (including Windows NT until Service Pack 4) will crash when presented with forged fragments that have overlapping offsets.

Defense

Apply the latest patches, service packs and hotfixes for your operating system. If you run Windows NT and must remain on Service Pack 3, apply the teardrop hotfix. For other operating systems, check with your vendor for a patch to this problem. Configure firewalls to reassemble fragments rather than forwarding them; most firewall TCP/IP implementations should be hardened against this attack.

UDP Floods

Various spoofing attacks exploit simple TCP/IP services like Chargen and Echo to transmit useless bandwidth wasting data.

Warning

Attack Status: These attacks are most easily defended against by not running the unnecessary services they exploit.

Profile

UDP floods are extremely simple: By forging a UDP connection to the Chargen service running on one host that has the reply address of a host running the Echo service, a hacker can create a useless stream of data flowing between the two hosts. Creating enough of these streams causes a bandwidth denial of service.

Defense

Configure hosts to disable simple TCP/IP services that are not necessary. Configure routers to block UDP requests from the Internet to these services.

SYN Floods

SYN floods are simple attacks that exploit the connection mechanism of TCP.

Warning Attack Status: SYN Floods are easily defended against, but watch out for future variants.

Profile

The requesting client transmits a SYN message to the host service requesting service, and the receiving server responds with a SYN-ACK message accepting the connection. The client then responds with an ACK message, after which traffic can flow over the established bidirectional TCP connection.

When a server receives the initial SYN message, it typically creates a new process thread to handle the client connection requests. This process thread creation requires CPU compute time and allocates a certain amount of memory. When the TCP session is closed or after a fairly long time-out period, the server closes the TCP session which releases the memory used. The amount of memory and compute time therefore determines the number of simultaneous sessions a server can support.

SYN floods are bogus SYN messages sent to servers. How a SYN flood affects a victim computer depends upon its implementation of TCP/IP.

Some TCP/IP stack implementations are only capable of waiting for ACK messages from a limited number of computers because they have a limited memory buffer for connection establishments. If this buffer is filled with bogus connection initiations, the server will cease responding to further connection attempts until the attempts in the buffer time out.

In implementations that are not connection-establishment limited, SYN floods have a similar effect. Since the server doesn't know a legitimate SYN message from a bogus message, it allocates compute and memory resources to establishing a connection. By flooding the server with a large volume of requests, the server's maximum capacity can be used by these bogus and useless connection attempts.

Defense

The only defense against SYN floods is a good firewall that can recognize the characteristics of a SYN flood—numerous identical connection attempts coming from the same IP address. These firewalls can filter subsequent connections from the same host, thus eliminating these SYN floods.

Since the SYN flood source machine isn't looking for a response, there's no reason why the SYN flood attack software can't simply use randomly generated IP addresses in the source field. This sort of SYN flood can not be discerned from a simple high volume of traffic and is able to get past SYN flood filters.

Land Attack

The Land attack is a variation of the SYN flood that can cause unhardened TCP/IP implementations to "chase their tails" in a never-ending connection-establishment loop.

Warning Attack Status: Land attack is already obsolete and easily defended against.

Profile

In the Land attack, a specially crafted SYN packet is transmitted to a server host. Both the source and destination addresses of the SYN packet are set to the server's IP address. This causes the recipient server to SYN-ACK to its own address, which it subsequently ACKs and establishes an empty connection. Each connection will remain until the server operating system times out the connection due to inactivity. Various operating systems respond to the Land attack differently—Windows NT becomes extremely slow for about five minutes. Many Unix implementations crash. Most vendors have supplied a fix for the Land attack.

Defense

Apply Land attack patches, hotfixes, or the latest service packs. Configure firewalls to drop any packets that have an internal source address inbound on the external interface. This always indicates a spoofed address, and dropping these packets should be default policy in every firewall.

The following IP addresses are illegal on the Internet and should always be filtered in or out of your external interface:

- 10 domain
- 127 domain
- 192.168 domain
- 172.16 through 172.31 domain

In addition, you should filter your own assigned IP addresses.

Smurf Attack

The Smurf attack is an extremely effective denial-of-service attack based on the direct broadcast addressing feature of IP that allows a host to transmit data to all hosts in its subnet.

Warning Attack Status: Smurf attacks can be filtered by firewalls, but they may still waste the bandwidth of your Internet connection because Smurf attacks outside your network can affect you.

Profile

A simple Smurf attack proceeds by flooding a victim host with ICMP echo request (ping) packets that have the reply address set to the broadcast address of the victim's network. This causes all the hosts in the network to reply to the ICMP echo request, thereby generating even more

traffic—typically one to two orders of magnitude more traffic than the initial ping flood.

A more complex Smurf attack proceeds as above but with the source of the echo request set to a third-party victim, which will receive all the echo requests generated by the targeted subnet of hosts. This attack is useful to hackers because they can use a relatively slow link like a modem to cause an avalanche of ping traffic to be sent to any location on the Internet. In this way, a hacker with a slower link than his ultimate victim can still flood the ultimate victim's pipe by smurfing a higher speed network than that of the ultimate victim.

Defense

To prevent hackers from exploiting your network to attack others, shut off the broadcast addressing feature of your external router or firewall. To prevent being the ultimate victim of a Smurf attack, configure your firewall to drop ICMP ping messages. If a high-speed provider (like your ISP) has been successfully smurfed and your network is the victim, there's nothing you can do about relieving the congestion that the ICMP traffic will cause, even if you filter it out on your end. Contact your ISP about their firewall policy if this is a concern.

Note Check out <http://www.powertech.no/smurf> to determine if your ISP is vulnerable to Smurf attacks.

Fraggle Attack

The Fraggle attack is a simple modification of the Smurf attack, which uses UDP echo messages rather than ICMP.

Warning Attack Status: Like its sibling the Smurf, the Fraggle attack can be filtered but may still consume network bandwidth in areas outside your control.

Profile

Fraggle is a simple rewrite of the Smurf attack that uses UDP rather than ICMP. This allows the attack to pass through firewalls that only filter ICMP.

Defense

Filter out UDP Echo messages at the firewall.

E-mail Bombs

Hackers can flood an e-mail server by repeatedly sending the same large e-mail file to your e-mail server.

Warning Attack Status: E-mail bombing and its relative, spamming, are fairly easy to filter, but they waste bandwidth outside your zone of control.

Profile

E-mail bombs are one of the oldest annoyance attacks around. By setting up a machine to constantly transmit e-mail to the same address, the attacker can use up bandwidth on the receiver's network.

This attack isn't all that serious, considering the fact that it requires nearly as much bandwidth on the transmitter's end as it does on the receiver's, which makes true denial of service difficult. E-mail bombs are also easy to trace, unless they've been transmitted through a mail host that strips headers.

Defense

Configure mail hosts to automatically delete excessive or duplicate messages from the same host.

Malformed Message Attacks

Many services on various operating systems will crash if they receive malformed messages because the services do not adequately check for errors in messages before processing them.

Warning Attack Status: Malformed message attacks will always be around. The number and variety of them makes it impossible to make general statements about vulnerability.

Profile

All operating systems have susceptibilities (known or unknown) to various message malformation problems. Variations include:

- E-mail buffer overruns from malformed e-mail messages
- Web services that can be exploited by providing exceptionally long URLs
- Crashing RPC services by transmitting random data to them

Defense

Stay up-to-date on the latest vulnerabilities reported by your software vendor. Apply patches and hotfixes to production machines as soon as you're certain that they are stable.

Exploitation Attacks

Exploitation attacks are those attacks that attempt to take direct control of a machine. Here's a list of some of the most common:

- TCP/IP connection hijacking
- Layer-2 connection hijacking
- Password guessing
- Trojan horses
- Buffer overruns

TCP/IP Connection Hijacking

TCP/IP connection hijacking exploits the fundamental non-randomness of TCP sequence generation algorithms to insert packets into an existing TCP/IP connection in order to "hijack" the connection.

Warning Attack Status: This attack is hibernating. Once common because simple pseudorandom

number generators were used to create sequence numbers, most modern TCP/IP stacks now have highly sophisticated sequence number generators. However, they are not yet truly random, so there's no reason to believe that an as-yet unknown algorithm can't make this attack common again.

Profile

TCP streams use pseudo-randomly generated sequence numbers to order TCP packets and ensure that the stream is reconstructed in the correct order at the remote end. If a hacker can predict the next correct sequence number, she can insert her own packets that will be received as if they were the correct packets. Subsequent packets from the original host will be discarded as spurious retransmissions or denial-of-service attacks that should be dropped.

The problem is in predicting the next correct sequence number. The difficulty in determining the next valid sequence number depends entirely upon the quality of the pseudorandom number generation (PRNG) algorithm used by the host computer to generate the sequence numbers for the TCP stack.

Computers are terrible at generating random numbers; the same code run twice on any two computers will always generate the same set of numbers. Therefore, PRNGs must rely on some external data as a source for "randomness." Early on, some PRNGs used sources as simple as the real-time clock as external data, but this source isn't at all random and is easily predicted. Better sources include timings between keystrokes or the frequency of other I/O operations, but these only work well on computers that have humans using them. Without the human element, these operations are all just as predictably ordered as anything else in a computer. So, for devices, there is no truly "good" source of random numbers for TCP sequence generation.

RFC 1948 suggests using the host's IP address and the source and destination ports along with a fixed truly random "key" number stored on the host to generate random numbers. As long as this key remains secret, this mechanism should be sufficient to ensure random sequence numbers.

Should be. The remaining problem is that no matter how random the initial number is, subsequent numbers are generated by a PRNG. All PRNGs, no matter how random the sequence "looks," contain predilections toward certain sequences (this predilection is called *strange attraction*, meaning that for some strange reason, the numbers are attracted to a certain order). These strange attractors can be graphed visually in two or three dimensions to show their non-randomness. While predicting the next random number is still very difficult, predicting a large set of numbers of which one might be the next random number is relatively easy. And if you remember that the remote host will simply drop "wrong" sequence numbers, the hacker can push a very large set of guesses at the remote host and need only hit on one to hijack the stream. Considering that a hacker could easily push 5,000 guesses down the pipe without straining the connection on a standard T1, it's conceivable that the entire set of likely sequence numbers could be used to subvert the connection. Again, this depends upon the quality of the PRNG.

Note TCP sequence prediction is how Kevin Mitnick (see <http://www.2600.com/>) performed many of his most famous hacks.

So how likely is it that your devices and computers could be exploited by a TCP hijacking attempt? Linux uses a very good PRNG, which makes sequence number generation highly unlikely. OpenBSD 2.9 has a PRNG that makes prediction impossible, but there's a slight chance that a hacker could predict sequence numbers in OpenBSD 2.8 or below. Windows 2000, IRIX, Mac OS X, and NT4 SP6a are somewhat susceptible to sequence number prediction, and NT4 SP3,

Windows 9x/Me, Mac OS 9, most commercial variants of Unix, and nearly every network device are so flawed that a competent hacker would be able to predict sequence numbers.

Tip <http://razor.bindview.com/publish/papers/tcpseq.html> has an excellent and practical analysis of sequence number prediction for major operating systems.

Defense

You can use a socket redirector (such as `redir`, available for various versions of UNIX) and a host OS that is invulnerable to prediction (such as OpenBSD 2.9 or Linux) to redirect all TCP streams (even if you use a 1:1 redirection) before they go to your network assets. Redirectors receive and then regenerate TCP streams—they are generic proxies and work with any protocol that can be NATed. NATs won't defend against this attack because they don't actually regenerate the sequence numbers.

Other than that, it's not possible to defend against this sort of attack. Once a PRNG has been reduced and encoded into an attack script, it's up to the original system vendor to patch the PRNG to provide increased randomness in its sequence numbers.

Layer-2 Connection Hijacking

Layer-2 connection hijacking exploits the nature of Layer-2 switches to perform a man-in-the-middle attack.

Warning Attack Status: Layer-2 connection hijacking is brand new and as-yet uncommon. It requires Layer 2 access to the network, which would be difficult in all but one circumstance: collocation at an ISP facility.

Profile

Layer-2 connection hijacking is a nifty little exploit that exploits assumptions that were made about the Ethernet ARP protocol that have since changed in the evolution of Ethernet.

ARP, Address Resolution Protocol, is used to map Ethernet MAC addresses to IP addresses. When a host comes online, it makes IP broadcasts to all hosts that cache the computer's MAC address so that other hosts can record the MAC address in their individual ARP tables and thus direct Ethernet packets to a specific host, given that host's IP address.

Duplicate IP addresses that would be used by a hacker to spoof an IP address are detected by the host that "owns" an IP address. If it sees a broadcast from another machine on the network that claims its IP address with another MAC address, it will complain, and the new host will be denied in the ARP tables of the receiving hosts. In a standard Ethernet broadcast environment, this works perfectly well.

Switches improve bandwidth utilization by routing traffic directly between the two MAC addresses involved in the conversation, rather than "broadcasting" the conversation to everyone. Switches maintain a MAC address table to determine which hosts are on which physical ports. But they don't track IP information because switches are not specific to IP; they can transport any Layer-3 networking protocol. This means that a multitude of hosts can communicate simultaneously through a switch without colliding with one another. This works extremely well—so well that switches have replaced hubs in most environments.

Enter the Layer–2 connection hijacking problem. When a host, say a web server, boots, it broadcasts its IP address and MAC address in an ARP packet to all hosts on the subnet, but most importantly to the default gateway. Enter the hacker's machine, which floods the MAC address of the default gateway with newer ARP packets that claim that the hacker's computer is the owner of the IP address. But because these packets aren't broadcast to all hosts on the switch, the original owner never hears them and so doesn't complain. The default gateway, assuming that a complaint would originate if the changed MAC address were illegitimate, records the new MAC address as the correct one for the web server's IP address. At this point, all traffic bound for the web server will now be transmitted to the MAC address of the hacker's computer.

The hacker, having the original ARP request of the web server, knows its MAC address. When it receives packets destined for the web server, it simply retransmits them to the web server's MAC address. Since this goes through the switch, no computers hear this connection except the hacker's computer and the web server. At this point, the hacker's computer has become a "bridge" between the default gateway and the web server, and is now able to monitor all data flowing between the two machines, including cryptographic key exchanges. From this point on, all information flowing between the web server and its default gateway is subject to whatever monitoring and reconstruction the hacker's computer is capable of performing. Linux is capable of performing all the Layer–2 bridging required for this exploit.

This exploit requires Layer–2 access to the network, so it's unlikely except in one environment: ISP Collocation. If you collocate machines at an ISP's facilities, your computers are connected to the same switch as potentially hundreds of other clients. Any of these clients could perpetrate this attack against your systems and monitor your data. While a hacker isn't likely to sign up for collocation service, they could easily use a *different* exploit to gain control of someone else's collocated server to then perform *this* exploit on yours. Any government or corporate interest with means and loose ethics could perpetrate this as well.

Defense

To defend against this attack, use hubs rather than switches between any of your public computers at a collocation facility. Or require that your ISP use Layer–3 routers rather than Layer–2 switches for connectivity. Considering that many network load balancers work better with hubs as well, you might consider making this your standard policy.

Password Guessing

Once a hacker has identified a host and found exploitable user account based services like NetBIOS, Telnet, or NFS, a successful password guess will provide control of the machine.

Warning Attack Status: Password guessing attacks are quite common, but are usually easily defended against. Watch out for remote control software like VNC or pcAnywhere that may be susceptible to high–speed automated attacks.

Profile

Most services are protected with account name and password combinations as their last line of defense. When a hacker finds an exploitable service running on a target machine, the hacker must still provide a valid account name and password in order to log in.

Automated password guessing software uses lists of common passwords, names, and words from the dictionary to attempt to guess high profile or important account names, such as the root user

password on Unix systems or the Administrator account on NT systems. The software typically takes a list of account names and a list of possible passwords to try and simply tries each account name with each password.

Hackers are using new "common password" lists to make these attacks faster. These lists are derived from the statistical analysis of stolen account information from exploited servers. By combining lists of stolen passwords and analyzing the lists for password frequency, hackers have created lists of passwords in order of how commonly they are used. This means that if any accounts on your network have relatively common passwords, hackers will get in, and quickly. We've used these lists to gain administrative access to servers in as little as 11 seconds over the Internet.

Defense

Use passwords that are difficult to guess, such as combinations of words and punctuation. Make sure exploitable services like NFS, NetBIOS, and Telnet are not exposed to the public. Establish lockout policies if the service supports it.

Trojan Horses

Hackers install software on exploited computers to further gain control of a system.

Warning Attack Status: Trojan horses remain an extremely dangerous, though difficult to accomplish, form of attack. However, many new worms and e-mail viruses exploit other vectors to get in and then use embedded Trojan horses to connect back out.

Profile

Trojan horses are programs that are surreptitiously installed on a target system either directly by a hacker, by a computer virus or worm, or by an unsuspecting user. Once installed, the Trojan horse either returns information to the hacker or provides direct access to the computer.

The most useful sorts of Trojan horses are called backdoors. These programs provide a mechanism whereby the hacker can control the machine directly. Examples include maliciously designed programs like NetBus, BackOrifice, and BO2K, as well as benign programs that can be exploited to give control of a system like netcat, VNC, and pcAnywhere. Ideal backdoors are small and quickly installable, and they run transparently.

The term "Trojan horse" doesn't apply to a specific type of software, it applies to how the software is used. There's nothing special about NetBus or BackOrifice that officially makes them Trojan horses, other than their typical use. They are just network-connected command prompt programs, the same way that VNC and pcAnywhere are network-connected GUI programs. The term Trojan horse applies to *how* the software is used, not why it was produced. Self-propagating Trojan horses are called *worms*.

Defense

Strip executable attachments from e-mail and avoid downloading untrusted software. Use network scanning software to constantly monitor listening TCP services on interior hosts—this will expose many known Trojan horses. Use proxy filters that block unknown protocols at the border.

Buffer Overruns

Buffer overruns are a class of attacks that exploit a specific weakness common in software.

Warning Attack Status: New buffer–overrun attacks are found all the time. IIS has been hit with so many new buffer overrun attacks in 2001 that many corporations are moving away from it as a service platform. Automated worms that exploit common IIS buffer overruns have swamped the net with scanning and copying activity as they search for victims and propagate.

Profile

Buffer overruns exploit the fact that most software allocates blocks of memory in fixed–size chunks to create a scratchpad area called a buffer, within which it processes inbound network information. Often these buffers are programmed to a fixed maximum size, or they are programmed to trust the message to correctly indicate its size.

Buffer overruns are caused when a message lies about its size or is deliberately longer than the allowed maximum length. For example, if a message says it's 240 bytes long, but it's actually 256 bytes long, the receiving service may allocate a buffer only 240 bytes long but then copy 256 bytes of information into that buffer. The 16 bytes of memory beyond the end of the buffer will be overwritten with whatever the last 16 bytes of the message contains. Hackers exploit these problems by including machine language code in the section of the message that is past the buffer end. Even more disturbing is the fact that software is often written in such a way that code execution begins after the end of the buffer location, thus allowing hackers to execute code in the security context of the running service.

By writing a short exploit to open a further security hole and postfixing that code to the buffer payload, hackers can gain control of the system.

Defense

Buffer overrun attacks are common and crop up all the time. Defend against them on public servers by staying up–to–date on the latest security bulletins for your operating system, or by using security proxies that can drop suspicious or malformed connections before they reach your server.

Information Gathering Attacks

Information gathering attacks are not exploits or denial–of–service attacks; they themselves do nothing harmful to the target. These attacks are used to provide information for further intrusion into a system. Hackers routinely employ these methods to obtain and identify targets. This section covers the following information gathering attacks:

- Various scanning techniques
- Architecture probes
- Exploiting information services

Address Scanning

Hackers use automated software to transmit ICMP echo messages to a range of target IP

addresses. Those that respond indicate that they exist.

Warning Attack Status: Address scanning is a common first sign of attack. Hackers often use address scanning and related scanning techniques to find hosts. Fortunately, address scanning is easy to filter.

Profile

By using automated software, hackers transmit ICMP echo messages across a very wide range of IP addresses (usually entire subnets). Those computers that respond become targets for further information-gathering attempts.

Defense

Filter out ICMP echo messages at the firewall.

Port Scanning

Hackers use automated software to establish TCP connections to various important ports on servers to determine if hosts exist and which exploitable services they run.

Warning Attack Status: Port scanning is widely used by hackers to identify targets, but it is easy to detect.

Profile

By using an automated tool, hackers connect to a sequence of TCP ports across a wide range of hosts. The scanning software reports those hosts with which it successfully established connections. Port scanning is the usual first step in an attack.

Defense

Software that can detect numerous connection attempts from the same host can be used to detect scanning. You can also use seduction servers operating on ports normally used by standard TCP/IP services to detect scanning. Many firewalls detect scanning and can be configured to automatically block scanning attempts.

Inverse Mapping

Scanning is becoming worthless for hackers since firewalls can filter ICMP echo messages, and scan detectors, or deception services, can detect scanning activity. Hackers have figured out how to use packet types that cannot be filtered to infer information about the interior of a network.

Warning Attack Status: Inverse mapping attacks are easy to filter out.

Profile

Inverse mapping works by inferring information from ICMP router replies. Hackers can infer which hosts exist by the lack of an ICMP "host unreachable" message when they transmit bogus response messages to hosts. These responses generate no reply from the host, but they also generate no error from the forwarding router. Those hosts for which no "host unreachable" messages are returned obviously exist. This information attack is a response to scan detectors.

Since normal scanning activity is easily detected by firewalls, hackers use common message types that firewalls cannot be configured to trigger on. These message types include:

- RESET messages.
- SYN-ACK messages, which would come from legitimate connection attempts to public servers from internal clients and are, therefore, extremely common.
- DNS response packets, which are replies to messages generated internally.

Defense

Stateful inspectors and true Network Address Translators automatically protect against this exploit, as do non-routing proxy servers. To be certain, filter ICMP host unreachable replies or configure your firewall to use Stealth mode, which stops replies to ICMP and unsuccessful TCP SYN messages.

Slow Scanning

Port scanning detectors can detect a high rate of similar connections going to numerous ports. Solution: slow down the scanning rate.

Warning Attack Status: This attack is a modification of the port-scanning technique. Its utility is dubious considering the time delay it causes.

Profile

Slow scanning methods rely upon the fact that firewalls and scan-detection software expect to see a high rate of connections from a single address to determine whether or not a scan is occurring. Scan detectors detect high rates by keeping track of the number of connections a specific host requests in a certain time frame (10 per second, for example). By scanning slower than the timeout rate of scan detectors (one attempt per second, for example), slow scanners avoid detection.

Unfortunately for hackers, the slow scan inherently takes a long time, so these attacks are only used when specific systems are attacked for a purpose.

Defense

Slow scans are very difficult for scan detectors to find. Seduction services are a more appropriate detector since they shouldn't be hit at all for legitimate traffic.

Architecture Probes

Hackers transmit garbled packets to host computers. By examining the responses, hackers may be able to determine the operating system running on the target machine.

Warning Attack Status: Currently rare, these attacks will become increasingly common.

Profile

Hackers examine the responses to bad packet transmissions from a target host using an automated tool that contains a database of known response types. Because no typical response definition exists, each operating system responds in a unique manner. By comparing unique responses to a database of known responses, hackers can often determine which operating system the target host

is running. Hackers can also use a simple statistical analysis of the TCP sequence numbers from any connection stream originating on your computer to "fingerprint" the TCP/IP stack on your machine. You can even tell which service pack an NT 4 server is running using this method, and it is impossible to prevent others from doing as well.

Defense

Assume hackers can determine which operating systems your public hosts run. Plan your defenses such that they do not rely upon security through obscurity. For example, you shouldn't assume a hacker can't tell you're running Windows NT Server on your machine because you've blocked identifying ports. You should still take all security measures to secure an operating system, even if you don't think a hacker knows which operating system it is.

DNS Zone Transfers

Hackers can transfer name information from your DNS server to identify internal hosts.

Warning Attack Status: This attack does not constitute a denial of service or exploitation itself and is easily defended against.

Profile

The DNS protocol does not perform authentication for transfers or informative updates. This makes the protocol exploitable in a number of different ways. Hackers can perform a zone transfer to get the names and internal IP addresses of all your hosts in a single operation if you maintain a public DNS server.

Defense

Smaller organizations should not run their own DNS servers. Use DNS servers that support split DNS to ensure internal names and addresses remain private. Filter zone transfer requests at the firewall. Use internal DNS servers not available from the Internet for inside names and separate public DNS servers for public names to prevent this attack and to prevent hackers from querying your DNS server to determine the internal IP ranges used on your network.

Finger

The Finger protocol can provide enough information about users to allow hackers to guess usernames or passwords. Finger was commonly employed in the early days of the Internet as a way to lookup e-mail addresses.

Profile

Hackers use the Finger command to probe a Finger server for information about the users of a system. Often, account names and even clues that yield passwords can be obtained.

Defense

This common attack made the Finger protocol obsolete. Nobody expects to get useful information from it anymore, so simply disable it. Block and log Finger connection attempts at your firewall.

LDAP

The Lightweight Directory Access Protocol is yet another information leaking service. By providing LDAP information to the public, which might include valuable clues into the nature of your network and its users, you provide a wealth of information to hackers.

Profile

Hackers use the LDAP protocol to glean information about systems inside networks and their users.

Defense

Block and log LDAP probes into your inside network. If you provide LDAP services on public machines, allow LDAP into your DMZ. Consider a "split LDAP" policy where you provide full LDAP services inside your network but only a limited set of information publicly.

SNMP Leakage

Simple Network Management Protocol was designed to automatically provide the configuration details of network devices. As such, "leaky" devices in a DMZ or on the public side of your network can provide a wealth of information about the interior of your network.

Warning Attack Status: Common, but usually low risk. Some devices that can be controlled via SNMP would be especially susceptible.

Profile

Nearly every type of network device, from hubs to switches to routers to servers, can be configured to provide SNMP configuration and management information. Interfaces like DSL adapters and cable modems are frequently SNMP configurable, as are many firewalls. Because of the ubiquitous nature of SNMP, it is frequently overlooked on devices that exist outside the public firewall, providing a source of information about your network and the possibility that a device could be remotely managed by a hacker. An example of this is a common brand of VPN controller that must be placed in parallel with the firewall and which exposes its SNMP information to the public interface so that it can be controlled by the service provider. These boxes leak a tremendous amount of information about the structure of the internal network, including all internal IP addresses and MAC addresses. This information can tell a hacker exactly how attack the network, such as to source-route through a NAT (an exploit that the same brand of VPN device was vulnerable to until the most recent firmware patch, by the way).

Defense

Disable SNMP on every device that exists outside your firewall—even if a service provider says they need it. It's not their network; it's yours. If they refuse to reconfigure the devices, drop their service and go with someone who really understands security.

Disinformation Attacks

Disinformation deceives the attack target to plant incorrect information. This information paves the way for future attacks. This section identifies three disinformation attacks:

- DNS cache pollution
- Registrar usurpation
- Forged e-mail

DNS Cache Pollution

Hackers can provide bogus updates to DNS servers with incorrect IP addresses.

Warning Attack Status: DNS cache pollution is becoming increasingly difficult for hackers to accomplish due to strong security by major ISPs, but it is still not impossible.

Profile

Because DNS servers do not perform authentication when exchanging information with other name servers, hackers can insert incorrect information that would divert users to the hacker's own hosts.

Defense

Filter inbound DNS updates at your firewall; no external name server should update your internal server's knowledge of interior machines. Firewalls that support split DNS already support this line of defense.

Registrar Usurpation

Internet registries (also called Internet Network Information Centers) form the root of the DNS name system. Some registries have reported successfully forged update messages that have allowed hackers to redirect site traffic from the legitimate owners.

Warning Attack Status: Most registrars have changed their update policies to more carefully guard against these attacks.

Profile

Some Internet name registrars rely on e-mail messages for updates. They compare the e-mail address of the registered owner to incoming e-mail addresses containing update messages, and, if they match, the registrar performs the update. Because e-mail can be forged, these updates may be illegitimate.

Defense

It is not possible for end users to defend against these attacks. Most registrars have enacted more stringent update confirmation mechanisms, including reply-to e-mail and telephone confirmation.

Forged E-mail

Hackers can create e-mail that appears to be coming from anyone and requires a reply. In a variation of the attack, they can spoof the reply-to address as well, making the forgery undetectable.

Warning Attack Status: E-mail forgery is probably the most serious form of attack against your network, and it's extremely difficult to defend against.

Profile

Using techniques as simple as configuring an e-mail client with incorrect information, hackers can forge e-mail to your internal clients. By claiming to be from someone the client knows and trusts, this e-mail uses a psychological attack to induce the reader to return useful information or includes an installable Trojan horse or a link to a malicious website.

SMTP does not authenticate the identity of an e-mail sender, and many versions of e-mail programs do not log enough information to properly track the source of an e-mail message.

E-mail servers normally include a chain of transmission headers in their e-mail messages, but numerous e-mail servers are known to strip these headers. Since hackers know which servers strip e-mail headers, they use them to make their attacks anonymous. Laundering IP addresses through a poorly configured proxy server can also make e-mail untraceable.

Tip Reading a list of servers known to strip e-mail headers (called "anonymizers" by hackers) is like reading a Who's Who of the .gov (government) domain. Why does the government run so many servers that strip headers? So it can keep track of all the machines from which requests to send striped-header e-mail originate.

Defense

The only true defense against e-mail forgery is user awareness; make sure your users understand that e-mail forgery is possible and constitutes a likely attack mechanism in well-defended networks.

You can use S/MIME enabled e-mail clients and install personal encryption certificates to sign e-mail from all internal users. Unsigned e-mail should be considered potentially suspect. Filter executable attachments out of e-mail at the firewall. Consider using PGP or certificates to authenticate your e-mail. Unfortunately, authenticated e-mail hasn't really caught on, so you can't expect to be able to authenticate the e-mail you receive except within your own organization.

Case Study: As You Sow, So Shall You Reap

About a year ago, I visited an old high school friend who had recently become very adept at hacking in his spare time. Eager to show off the spoils of his newfound pastime, he pulled up movies, music, and software he'd gotten for free from the Internet.

Reluctant to moralize, I sat back and took the opportunity to immerse myself in the psyche of an active hacker. While we talked and watched the random gyrations of WinAmp (an MP3 music player that creates random psychedelic patterns on the screen that are synchronized to the music), a command-prompt window suddenly popped up.

We both leaned forward and watched incredulously as the following letters appeared one at a time, as if typed by unseen hands:

C:\>Format c:

The cursor paused at the end of the command line. My friend rushed to his machine and immediately closed the command prompt. We expressed shock at what we both knew was a Trojan horse attack, and then I immediately set about to defend and respond. Having never actually been present during a successful intrusion, the opportunity excited me tremendously.

My first step was to issue the `netstat -a` command. We quickly identified an unknown listening service on port 12345.

Since I recognized the common listening ports for a Windows computer, I knew this one was unusual. I recorded the remote IP address of the computer attached to it.

The mysterious command window appeared again, so I immediately shut it again. I then pulled up a port scanner and scanned the IP address of the remote computer. The results of the port scan revealed a listening service on port 135—the sure sign of a Windows NT computer. The computer wasn't running other proxy services, so I was comfortable that the hacker hadn't laundered his (I'm assuming the hacker was male) IP address.

To end the attack, I took a chance that the hacker hadn't updated to the latest service pack and telnetted to his RPC port. Typing random data into this port caused a blue-screen denial-of-service attack in early Windows NT 4 service releases. Sure enough, a subsequent port scan revealed that the attacker's machine was no longer responding.

We then set about searching for the Trojan horse program on my friend's computer. After some research and looking, we found a copy of the NetBus Trojan horse. It had been installed by one of the pieces of pirated software my friend had downloaded from a hacker's website.

Chapter 13: Security Utilities

Overview

There are a number of important security functions that firewalls do not fulfill, such as vulnerability analysis and disk encryption. This chapter covers the software utilities that we've found to be especially useful and a good value in our security practice.

You have to be cautious when using multiple small-scale tools to provide security; they are not integrated, they are usually not robust, and they invariably rely upon the strength of the operating system's TCP/IP stack, which could be vulnerable to denial-of-service attacks and other problems.

The lack of integration among numerous security utilities causes the "Swiss cheese" effect, where a combination of separated tools leave holes through your bastion host. Multiple tools can't prevent problems that integrated solutions can. For example, using a separate NAT and proxy is an either/or solution: you can either proxy a protocol or you can use NAT to pass it to the Internet. Good firewalls can both proxy and perform Network Address Translation on connections flowing through, thus providing stronger security.

Some security is always better than no security (as long as you aren't lulled into complacency by thinking your solution is stronger than it is). Given the budget realities in many small organizations, we think it's better to do what you can within the limits of your resources.

Note The security tools presented in this chapter are not replacements for firewalls, and they should not be used instead of firewalls.

This chapter presents security utilities for Windows or Unix; they hold by far the most market share among bastion hosts. This chapter should not be considered an exhaustive review of small security utilities—rather, it's a mixed tool bag of software we know works well over the long term.

Software You Already Have

You should be aware of what the operating system utilities you already have can do for you. Windows and Unix come with a wide range of security-related tools that you can use to monitor your network. Other operating systems like NetWare and the Macintosh OS are considerably more limited in this respect.

Unix/Linux Utilities

The first step to securing your Unix or Linux computer is to know what's going on inside it. Most Unix services are quite happy to tell you everything you'd like to know about their operation in the syslog, which you can either check by hand or scan automatically if you like. Top will show you there's something unusual running and eating up your system resources, netstat will identify which ports are open and who's connecting to them, find and grep are your automated file searching tools, and diff will show you differences between files.

syslog

There is a file to which Unix services append status messages (using the syslog daemon) when they start up, when they fail, and when they have denied access to a user due to a security

violation. This file is /var/log/messages in later versions of Linux; other versions of Unix may give it another name, but it serves the same purpose—and it does no good if you don't review it. An example snippet of a syslog follows, in which you can determine that someone is attempting and failing to use the SMTP service (which has been configured to use password authentication to foil spammers using it as an open relay), and someone else has successfully used the FTP service.

```
Oct 19 09:31:04 blahtech smtp(pam_unix)[4878]: check pass; user
unknown

Oct 19 09:31:04 blahtech smtp(pam_unix)[4878]: authentication
failure; logname= uid=0 euid=0 tty= ruser= rhost=

Oct 19 09:31:06 blahtech server cx29073-a.zone3.townsville.home.com
[24.0.99.202] cmd read[4878]: NOQUEUE: cx29073-
a.zone3.townsville.home.com [24.0.99.202] did not issue MAIL/EXPN/
VRFY/ETRN during connection to MTA

Oct 19 09:31:12 blahtech smtp(pam_unix)[4879]: check pass; user
unknown

Oct 19 09:31:12 blahtech smtp(pam_unix)[4879]: authentication
failure; logname= uid=0 euid=0 tty= ruser= rhost=

Oct 19 09:31:14 blahtech server cx29073-a.zone3.townsville.home.com
[24.0.99.202] cmd read[4879]: NOQUEUE: cx29073-
a.zone3.townsville.home.com [24.0.99.202] did not issue MAIL/EXPN/
VRFY/ETRN during connection to MTA

Oct 19 10:50:16 blahtech ftpd[5041]: FTP LOGIN FROM
associatedproducts.com [11.221.232.90], phillip

Oct 19 10:51:28 blahtech ftpd[5041]: FTP session closed
```

top

When your Unix mail server slows down, are you being hacked or is the machine merely processing a lot of requests? You can find out by using the top command, which can show you the most resource-intensive processes running in the computer, sorted by percentage of CPU used or percentage of memory used.

```
 8:05am up 22 days, 21:44,  5 users,  load average: 0.00, 0.00,
0.00

68 processes: 67 sleeping, 1 running, 0 zombie, 0 stopped

CPU states:  0.5% user,  0.1% system,  0.0% nice, 99.2% idle

Mem:  384408K av,  373068K used,   11340K free,         0K shrd,
49524K buff

Swap: 1052344K av,         0K used, 1052344K free
157720K cached


  PID USER      PRI  NI  SIZE  RSS SHARE STAT  %CPU  %MEM   TIME COMMAND
16581 root         16   0  1056 1056   836 R     0.3   0.2   0:00 top
      1 root         8    0   544   544   472 S     0.0   0.1   0:04 init
      2 root         8    0     0     0     0 SW    0.0   0.0   0:00 keventd
```

3 root idled	9	0	0	0	0 SW	0.0	0.0	0:00	kapm-
4 root	9	0	0	0	0 SW	0.0	0.0	0:00	kswapd
5 root kreclaimd	9	0	0	0	0 SW	0.0	0.0	0:00	
6 root	9	0	0	0	0 SW	0.0	0.0	0:00	b 0:00 bdfush
7 root kupdated	9	0	0	0	0 SW	0.0	0.0	0:00	
8 root mdrecoveryd	-1	-20	0	0	0 SW<	0.0	0.0	0:00	
73 root	9	0	0	0	0 SW	0.0	0.0	0:00	khubd
601 root	9	0	600	600	500 S	0.0	0.1	0:27	syslogd
606 root	9	0	1104	1104	456 S	0.0	0.2	0:00	klogd
620 rpc	9	0	596	596	504 S	0.0	0.1	0:00	portmap
635 rpcuser rpc.statd	9	0	772	772	668 S	0.0	0.2	0:00	
759 root	8	0	532	532	464 S	0.0	0.1	0:00	apmd
810 root automount	9	0	648	648	544 S	0.0	0.1	0:00	
822 daemon	9	0	584	584	508 S	0.0	0.1	0:00	atd

netstat

Which ports are currently open on your Unix server? You could use a port scan package to find out, but the operating system itself will tell you and also inform you of the active connections to those ports. Netstat is the command to show open and active ports in most versions of Unix. An example output of netstat -l follows:

```
[root@anonymous /root]# netstat -l
```

Active Internet connections (only servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:exec	*:*	LISTEN
tcp	0	0	*:32768	*:*	LISTEN
tcp	0	0	*:5801	*:*	LISTEN
tcp	0	0	*:5802	*:*	LISTEN
tcp	0	0	*:5901	*:*	LISTEN
tcp	0	0	*:5902	*:*	LISTEN
tcp	0	0	*:pop3	*:*	LISTEN

```

tcp      0      0 *:sunrpc      *:*           LISTEN
tcp      0      0 *:webcache    *:*           LISTEN
tcp      0      0 *:x11         *:*           LISTEN
tcp      0      0 *:http        *:*           LISTEN
tcp      0      0 *:6001        *:*           LISTEN
tcp      0      0 *:6002        *:*           LISTEN
tcp      0      0 *:ftp         *:*           LISTEN
tcp      0      0 *:ssh         *:*           LISTEN
tcp      0      0 *:smtp        *:*           LISTEN
tcp      0      0 *:https       *:*           LISTEN
tcp      0      0 *:3389        *:*           LISTEN
udp      0      0 *:32768       *:*
udp      0      0 *:811         *:*
udp      0      0 *:sunrpc      *:*

```

Active UNIX domain sockets (only servers)

Proto	RefCnt	Flags	Type	State	I-Node	Path
unix	2	[ACC]	STREAM	LISTENING	1197	/dev/gpmctl
unix	2	[ACC]	STREAM	LISTENING	3380	/tmp/.X11-unix/X0
unix	2	[ACC]	STREAM	LISTENING	8390	/tmp/.X11-unix/X1
unix	2	[ACC]	STREAM	LISTENING	39835	/tmp/.X11-unix/X2
unix	2	[ACC]	STREAM	LISTENING	1237	/tmp/.font-unix/
fs7100						

When examining the open ports on your servers, you should look for anomalous behavior that could be indicative of a back door or Trojan horse. Why, for example, would the Linux server in the above example be listening on port 3389, which is the Windows Terminal Services port? Since Linux does not provide Terminal Services support, that open port is suspicious and requires further investigation.

IPChains/ipf

When you set up a firewall with Linux and IPChains (or BSD and ipf), you instruct the computer on which packets to allow through and which ones to drop. To keep your network secure you need more than that—you also need to know when your network is under attack and when those rules are being violated. You can configure IPChains to log whenever a rule is activated (such as denying ICMP packets) to the syslog. You can then browse the log for entries such as:

```

Packet log: input DENY eth0 PROTO=17 192.168.2.1:53 192.168.1.1:1025
L=34 S=0x00 I=18 F=0x0000 T=254

```


The various fields are broken down as follows:

1. The input chain originated the log message.
2. Deny indicates that the packet was denied.
3. The packet came in on eth0.
4. PROTO=17 announces that this was a UDP packet.
5. 192.168.2.1:53 shows the source IP and port numbers (or ICMP type) of the packet.
6. 192.168.1.1:1025 is the packet's destination IP and port numbers (or ICMP type).
7. L=34 shows the length of the packet to be 34 bytes.
8. The Type of Service field S=0x00 indicates that no TOS bits were set.
9. I=18 is the IP Identifier.
10. F=0x0000 indicates that this is not a fragment.
11. The Time To Live field T=254 is decremented at every router hop for the packet.

To include logging of a rule in IPChains, you include the `-l` flag with the specific rule you want to log the activation of.

Windows Utilities

Like Unix, Windows comes with a number of small tools to assist you in security administration. The Windows built-in tools aren't quite as comprehensive as the set that comes with Unix, so you'll probably wind up supplementing them with third-party utilities.

Windows Event Viewer

The Windows Event Viewer utility displays system logs and allows you to filter the display to show certain types of events. The security log includes security violations, such as account lockouts. Any auditing you turn on with the User Manager's Audit Policy dialog box also causes events to be written to these logs. You have to enable security logging in the User Manager for anything to happen, since all auditing is disabled by default.

Event Viewer is located in the Administrative Tools menu under the Start menu. When you first run the Event Viewer utility, a list of events in the system log is displayed. An icon indicating its significance precedes each entry. Figure 13.1 shows the Windows Event Viewer.

Date	Time	Source	Category	Event	User	Computer
7/28/99	1:39:17 AM	Security	Logon/Logoff	528	IUSR_CX9246	CX924688-A
7/27/99	8:33:34 PM	Security	Logon/Logoff	538	IUSR_CX9246	CX924688-A
7/27/99	8:02:02 PM	Security	Logon/Logoff	528	IUSR_CX9246	CX924688-A
7/27/99	7:43:19 PM	Security	Logon/Logoff	538	netropolis	CX924688-A
7/27/99	7:28:18 PM	Security	Logon/Logoff	528	netropolis	CX924688-A
7/27/99	7:03:33 PM	Security	Logon/Logoff	538	IUSR_CX9246	CX924688-A
7/27/99	6:43:41 PM	Security	Privilege Use	577	netropolis	CX924688-A
7/27/99	6:26:09 PM	Security	Logon/Logoff	528	IUSR_CX9246	CX924688-A
7/27/99	6:11:11 PM	Security	Logon/Logoff	538	netropolis	CX924688-A
7/27/99	6:03:33 PM	Security	Logon/Logoff	538	IUSR_CX9246	CX924688-A
7/27/99	6:00:14 PM	Security	Logon/Logoff	528	netropolis	CX924688-A
7/27/99	5:59:03 PM	Security	Privilege Use	577	netropolis	CX924688-A
7/27/99	5:56:12 PM	Security	Privilege Use	577	netropolis	CX924688-A
7/27/99	5:46:43 PM	Security	Logon/Logoff	528	IUSR_CX9246	CX924688-A
7/27/99	5:46:43 PM	Security	System Event	515	SYSTEM	CX924688-A
7/27/99	4:49:14 PM	Security	Logon/Logoff	538	netropolis	CX924688-A
7/27/99	4:36:43 PM	Security	Logon/Logoff	538	netropolis	CX924688-A
7/27/99	4:34:03 PM	Security	Logon/Logoff	528	netropolis	CX924688-A
7/27/99	4:19:12 PM	Security	Logon/Logoff	528	netropolis	CX924688-A
7/27/99	4:19:00 PM	Security	System Event	515	SYSTEM	CX924688-A
7/27/99	4:18:29 PM	Security	System Event	515	SYSTEM	CX924688-A
7/27/99	4:18:29 PM	Security	System Event	515	SYSTEM	CX924688-A
7/27/99	4:18:24 PM	Security	System Event	515	SYSTEM	CX924688-A
7/27/99	4:18:22 PM	Security	System Event	515	SYSTEM	CX924688-A

Figure 13.1: The Windows Event Viewer showing the security log
Windows Network Monitor

The Windows Network Monitor allows you to capture network packets and display information about them in extremely detailed form and with the protocols clarified. This can be a useful tool for monitoring the usage of the network, as well as searching for specific packets to track down security problems.

Before Network Monitor can be used, you must install the Network Monitor Agent on one or more computers and the Network Monitor Tools and Agent service on the computer from which you will do the monitoring.

To install these services under Windows NT, open the Network Control Panel. Select the Services tab, then click Add to add a service. Select the appropriate service (either Network Monitor Agent or Network Monitor Tools and Agent) and click OK. You must then restart the computer. Figure 13.2 shows the Windows Network Monitor performing a packet analysis.

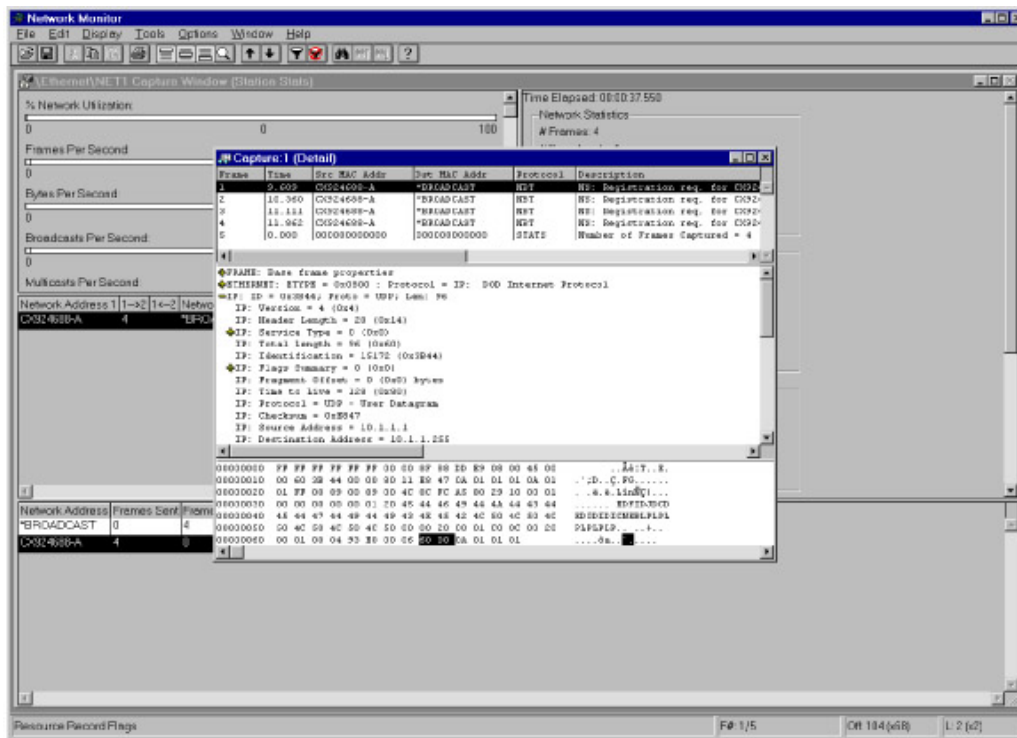


Figure 13.2: The Windows Network Monitor

The version of Network Monitor included with Windows can only capture packets coming from and going to the current workstation, which limits its use as a packet sniffer. This should be sufficient for a bastion host because all Internet traffic must go through the machine. The version included with Microsoft SMS (Systems Management Server) can capture all packets. This version may be more useful to perform diagnostics or just get an idea of the type of information a hacker could obtain by packet sniffing.

Performance Monitor

The Performance Monitor utility is one of the most useful Windows utilities for optimizing and monitoring performance and also has a few uses relating to security. To run this utility, select Performance Monitor from the Administrative Tools menu.

Performance Monitor deals with individual parameters, called *counters*, that relate to the system. You can use the View menu to switch between four methods of monitoring these counters:

- **Chart** Displays a running graph of selected counters.
- **Alert** Allows you to set minimum and maximum values for one or more parameters. You are alerted when a counter crosses one of these values.
- **Log** Creates a log of specified counters to a disk file.
- **Report** Allows you to create a customized report based on counter information.

There are a wide variety of counter categories available in Performance Monitor, each of which includes several counters. The Server category includes several counters dealing with security:

- **Errors Access Permissions** The number of times users have attempted to access files without proper access. A high number might indicate that a hacker is searching for accessible files.
- **Errors Logon** The number of invalid logon attempts. A dramatic increase in this counter may indicate that a hacker is attempting to guess a password, or running a program to try

passwords in succession.

- **Logon/sec** The number of logons per second. A change in this value may indicate that an intruder is repeatedly logging in and out, perhaps trying different default accounts.

Depending on your needs, other counters may be useful. For example, you can monitor network use and errors for the TCP/IP protocol using the TCP counter. Figure 13.3 shows the IP counter being examined for the number of datagrams received per second. Unfortunately, to use and alert on events using the Performance Monitor, you must be logged in, so it's typically most useful for investigating suspected hacking activity.

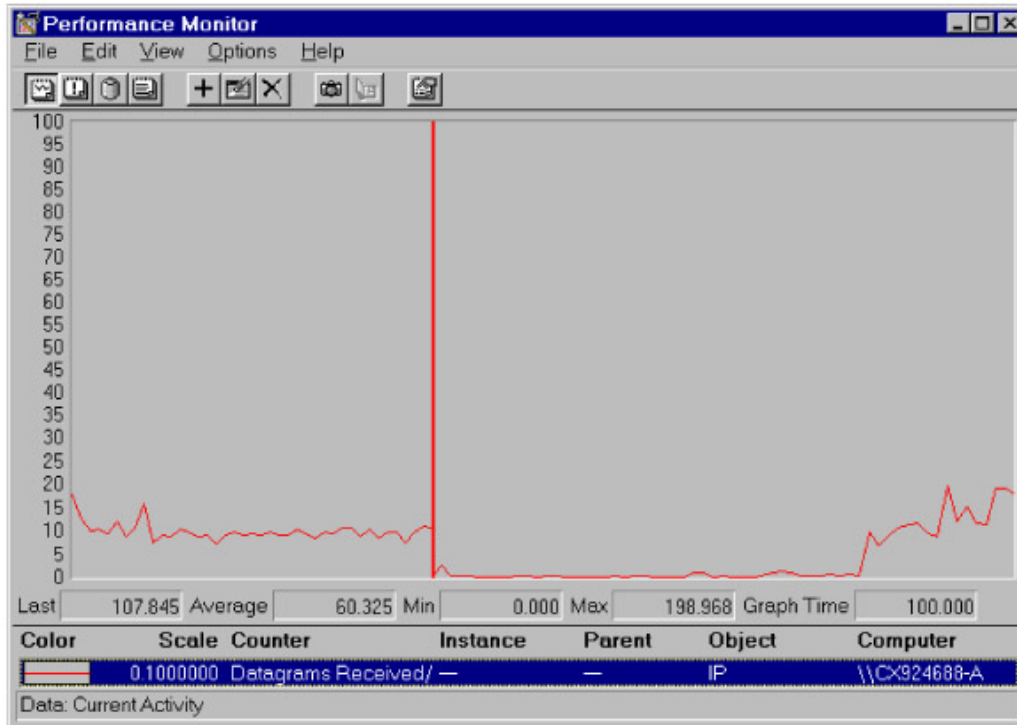


Figure 13.3: The Performance Monitor

CACLS

CACLS (Command-line Access Control Lists) is a Windows command-line utility that provides fine control over the assignment of permissions to files and directories. Since CACLS is a command-line utility, you can use it in batch files to perform mass changes to the permission structure of your drives.

Warning Be careful of the order in which you perform permission changes so that you don't deny yourself access before you can grant it! Always add new access permissions first, then delete inappropriate permissions.

When you type in CACLS with no command-line parameters, you'll see the instructions on how to use it, as shown in Figure 13.4.

```

C:\>cacls
Displays or modifies access control lists (ACLs) of files

CACLS filename [/T] [/E] [/C] [/G user:perm] [/R user [...]]
               [/P user:perm [...]] [/D user [...]]
  filename      Displays ACLs.
  /T            Changes ACLs of specified files in
                 the current directory and all subdirectories.
  /E            Edit ACL instead of replacing it.
  /C            Continue on access denied errors.
  /G user:perm  Grant specified user access rights.
                 Perm can be: R Read
                              C Change <write>
                              F Full control
  /R user       Revoke specified user's access rights (only valid with /E).
  /P user:perm  Replace specified user's access rights.
                 Perm can be: N None
                              R Read
                              C Change <write>
                              F Full control
  /D user       Deny specified user access.
Wildcards can be used to specify more than one file in a command.
You can specify more than one user in a command.

C:\>

```

Figure 13.4: The CACLS Permissions Tool

Most of the functionality of CACLS is described by the above commands, but there are a few things you should be aware of that are not immediately apparent:

- If you type CACLS with the name of a file or directory, the permissions for that file or directory are displayed. This is useful for showing exactly who has permission to what before you change anything. You can also save this information as a text file and later use a command interpreter like qbasic.exe to rebuild permissions.
- If you don't include the /E (edit) switch, the access control list is completely replaced. The access control list for an object is then created anew. You should usually include the /E switch.
- Accounts (user or group) with spaces in their identifiers (like "Domain Users") must be preceded by a single (") character, as shown here. Accounts without spaces do not require a quote.

```
CACLS . /E /R "NETROPOLIS\Domain Users"
```

- You can use a period to specify the current directory. Wildcards will show all files in the current directory.
- Use the /T operator to show or change permissions from the current location and in all subdirectories thereafter. The command below will record permissions for every file on your hard disk to a text file:

```
CACLS C:\*.* /T >C:\PERMIT.TXT
```

You can use batch files to control the functionality of CACLS; otherwise, there is little advantage to using it instead of the desktop explorer to change permissions on an NTFS volume.

Cross Platform Tools

Let's take a look at some tools that work with various platforms.

NSLOOKUP

NSLOOKUP (Name Server Lookup) is used to resolve Internet names to IP addresses. When you type in NSLOOKUP with no command-line parameters, and then type "?" at the greater-than prompt, you'll see instructions on how to use the command (this is true for most versions).

Typically, you'd use NSLOOKUP to resolve an IP address given a name, as in:

```
C:\>Nslookup www.ibm.com
```

This command will return the IP address of IBM's web server. Unlike most command-line utilities, NSLOOKUP has two modes of operation. If you type NSLOOKUP [name] and press return, NSLOOKUP will resolve the address using your default name server, return the results, and exit back to the command prompt. If you provide no parameters when you launch NSLOOKUP, it will run as a Unix-style command-line program, complete with its own prompt and list of commands. You can then use these various commands to perform all sorts of name server-related functions, such as changing your default name server. The use of most of these commands is esoteric and usually not necessary except for troubleshooting purposes.

PING

Ping is used to send ICMP echo messages (pings) to a remote host to determine if the host is available for further TCP/IP traffic. Ping is so often used to test for the existence of hosts and for the proper operation of network clients that it is commonly used as a verb among TCP/IP network integrators, as in "Did you ping the server?"

Ping is especially useful as a remote link status monitor. Using the continuous ping option (-t in Windows, -l in Linux (the default), and -s in most commercial Unix implementations), you can open a command shell and use the text output as a running second-by-second indicator of the connection status between the monitoring host and the monitored host. In Windows and X-Windows, you can open as many command shells as you want in order to monitor numerous hosts constantly. You can also use a scripting language to e-mail or page you when the link status changes.

You can control the various ICMP message parameters of the generated ping packet using the available parameters. Most of these options are used only during esoteric troubleshooting sessions.

Tip The success of a simple ping request can be used to determine exactly how a malfunctioning client is operating. If a ping to another computer succeeds, you know that the Physical layer, Data Link layer, and Network layer are all functioning correctly and that any communications problems you may be experiencing must be occurring in higher layers.

Hackers also use ping for various detrimental purposes. For instance, you can generate a Ping of Death from Windows 95 machines by typing the following:

```
C:\>Ping 10.1.1.1 -l 65510 -n 1000
```

The Ping of Death generates exceptionally large (>64K) and malformed ICMP echo requests that are transmitted to hosts with delicate TCP/IP stacks. Many TCP/IP implementations will crash when they can't decipher an ICMP message correctly, so these Ping-of-Death attacks can be used to crash some TCP/IP servers remotely. Windows NT was susceptible to various Ping-of-Death attacks prior to Service Pack 3, and may remain susceptible to some undiscovered attacks.

Note Microsoft eliminated the ability to generate a Ping Of Death in versions of Windows later than 95. This limitation doesn't affect other operating systems like Linux.

Telnet

Telnet is used to establish console user sessions with multi-user computers. Windows NT does not support multiple simultaneous users interactively, so it does not have a Telnet server. Windows 2000 does support Telnet for administrative services and allows two simultaneous users. The Telnet

client included with Windows is useful for establishing user console sessions on remote servers.

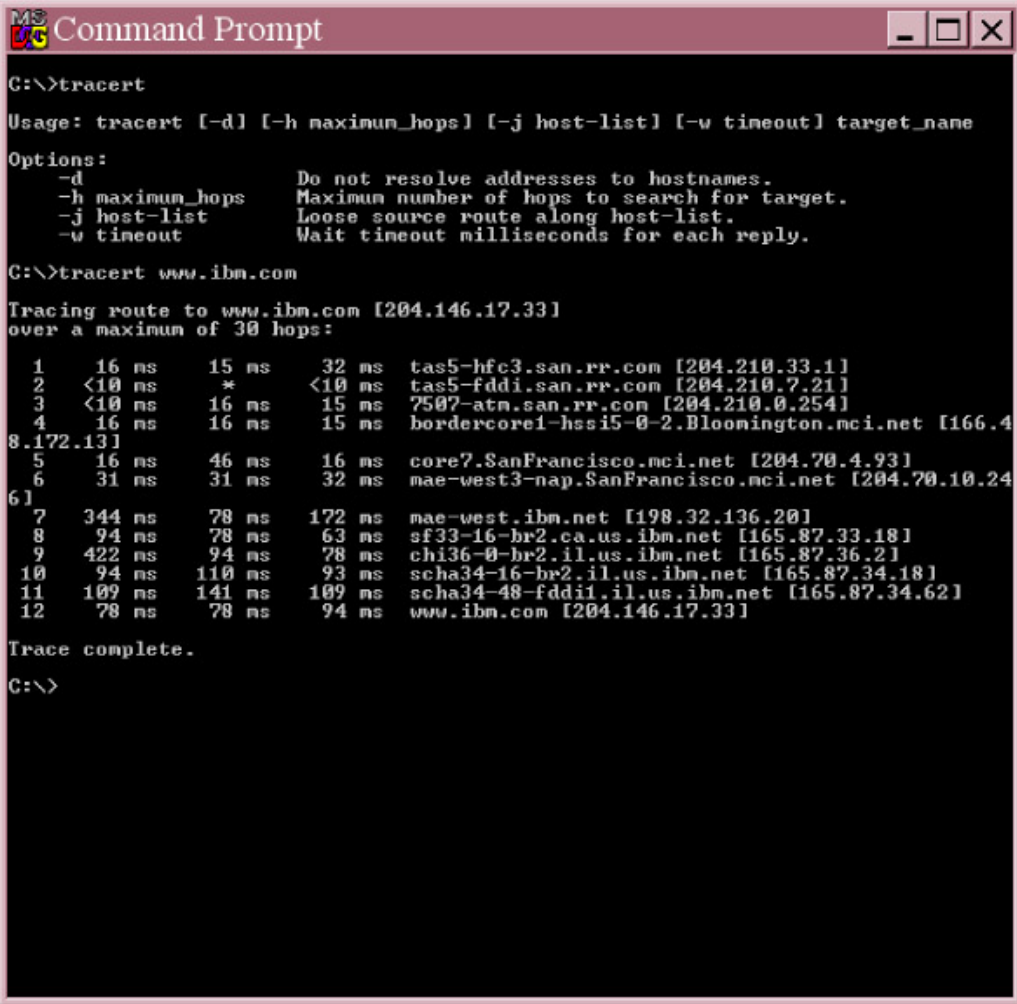
Telnet is especially useful to probe the presence and functionality of various Internet services. Since most classic Unix services provide plaintext responses, you can telnet to hosts and specify the service port to determine whether or not the service is running correctly. You can use Telnet to attach to the following services:

- Simple TCP/IP services like Echo, Daytime, Chargen, etc.
- Content services like HTTP and FTP
- POP3, SMTP, and NNTP

More advanced session layer services like CIFS/SMB and NFS don't provide human readable responses and may not connect to a Telnet client.

Tracert

Tracert (Trace Route, traceroute in Unix) is used to display the routers between two communicating Internet hosts. Figure 13.5 shows the command-line parameters of the tracert command and a sample route traced between two Internet hosts.



```
C:\>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] target_name

Options:
-d          Do not resolve addresses to hostnames.
-h maximum_hops  Maximum number of hops to search for target.
-j host-list  Loose source route along host-list.
-w timeout    Wait timeout milliseconds for each reply.

C:\>tracert www.ibm.com

Tracing route to www.ibm.com [204.146.17.33]
over a maximum of 30 hops:

  0  16 ns   15 ns   32 ms  tas5-hfc3.san.rr.com [204.210.33.11]
  1  <10 ns   *        <10 ms  tas5-fddi.san.rr.com [204.210.7.21]
  2  <10 ns   16 ns    15 ms  7507-atn.san.rr.com [204.210.0.254]
  3  16 ns    16 ns    15 ms  bordercore1-hssi5-0-2.Bloomington.mci.net [166.4
8.172.131]
  4  16 ns    46 ns    16 ms  core7.SanFrancisco.mci.net [204.70.4.93]
  5  31 ns    31 ns    32 ms  mae-west3-nap.SanFrancisco.mci.net [204.70.10.24
6]
  6  344 ns   78 ns   172 ms  mae-west.ibm.net [198.32.136.201]
  7  94 ns    78 ns    63 ms  sf33-16-br2.ca.us.ibm.net [165.87.33.18]
  8  422 ns   94 ns    78 ms  chi36-0-br2.il.us.ibm.net [165.87.36.21]
  9  94 ns   110 ns   93 ms  scha34-16-br2.il.us.ibm.net [165.87.34.18]
 10 109 ns   141 ns  109 ms  scha34-48-fddi1.il.us.ibm.net [165.87.34.62]
 11 78 ns    78 ns    94 ms  www.ibm.com [204.146.17.33]
 12

Trace complete.
C:\>
```

Figure 13.5: Tracert command sample trace

Since most Internet service providers use meaningful names on their router interfaces, you can often determine quite a bit about the route taken between two hosts. From the example shown in Figure 13.5, we can determine the following:

1. The first router interface is listed as tas5-hfc3.san.rr.com. Since we know from other hops on this service provider that they usually include the protocol for the interface, we can assume that hfc3 is some sort of physical port technology. It may stand for "High Frequency Cable Modem Channel 3," which would indicate that the end user is using cable modem technology to connect to the Internet. This line also indicates that the router is probably located in a city beginning with the letters "san"—which, following the common router practice of identifying a city by its three letter airport designator, would be San Diego, CA.
2. The next line is another interface on the same router—this time using fiber distributed data interface (FDDI), a 100Mbps token ring over optical fiber technology.
3. The next line indicates that the next higher router is using Asynchronous Transfer Mode (ATM) technology. This technology operates at bit rates varying from 25Mbps to 2200Mbps. As routers get closer to the Internet backbone, they should use increasingly faster data link technologies. This probably means that the ATM link is either 155, 622, or 2200Mbps.
4. The next line indicates that the next higher router uses High Speed Serial Interface (HSSI) technology, that it is located in Bloomington, and that it is operated by MCI.
5. The next line indicates that we've reached the Mae West Network Access Point (NAP) in San Francisco. This network access point is one of four commercial Internet exchanges in the country that forms the backbone of the Internet. So far, the route includes only very high-speed protocols and is relatively close to the Internet backbone—lucky user!
6. The next line indicates a switch from MCI's network to IBM's. This indicates that IBM has a presence on the Internet backbone directly, and would be a good candidate for consideration as an Internet service provider, as would MCI.
7. The next router is still in San Francisco and still on IBM's network.
8. The next router appears to be in Chicago (another NAP location).
9. The next router is in some city abbreviated "SCHA"—perhaps Schenectady, NY?
10. The next router indicates a shift down to FDDI.
11. The final hop is the destination server.

With practice and a strong knowledge of data link technologies, you can determine quite a bit about the identity of remote hosts on the Internet.

Security Analysis Tools

Security analysis tools scan target hosts for various known security vulnerabilities from another machine on the Internet. In essence, these tools provide one-stop-shopping to determine which known bugs or vulnerabilities your machines are susceptible to. Until you're completely familiar with Internet and operating system security, you should use these tools to discern where you need to shore up your host security.

Unfortunately, these tools operate from databases with a known problem—the databases can't find vulnerabilities that hackers don't already know about. This makes them more suitable for catch-up than strong security scanning. Exploits developed after the tool has been updated will be open. The only solution to that problem is to subscribe to e-mail vulnerability reports like SANS (<http://www.sans.com/>) and Microsoft Security Advisor (www.microsoft.com/security), and then do what they tell you to do.

SATAN

The Security Administrator's Tool for Analyzing Networks (SATAN) is an open-source Unix-based network security analysis tool that probes hosts on TCP/IP networks for security vulnerabilities. Although SATAN is primarily designed to find flaws in Unix hosts, it can point out problems in other

operating systems as well.

SATAN's original release caused a considerable stir because hackers immediately used it to probe and break into numerous public hosts. Since that time however, its ubiquitous availability has made many an Internet host safe from intrusion.

SATAN can be downloaded at www.fish.com/~zen/satan/satan.html. Other versions of SATAN (with cute names like SAINT and SANTA) also exist as separately maintained offshoots of the original tool.

WS-Ping

WS-Ping is the "best of breed" TCP/IP administration tool. It provides a number of other TCP/IP client services to help you administer your network. The very services that make WS-Ping useful to administrators also make it useful to hackers, so it's likely you'll run into hackers using it if you have monitoring software installed on your public serves.

Installing WS-Ping is simple—just run the included install program and the rest is done for you. You then launch it using the Start menu. WS-Ping provides the following services, which you can select by clicking the appropriately named tab:

- **Ping** Allows you to ping a host automatically with any sized packet for any duration.
- **Traceroute** Performs a TCP/IP trace route.
- **Lookup** Performs normal or reverse DNS name lookups.
- **Finger** Performs the finger function to get user details from Internet hosts.
- **Whois** Attaches to servers running the Whois services to resolve e-mail names.
- **LDAP** Allows you to attach to servers running the Lightweight Directory Access Protocol to glean account information.
- **Quote** Provides a Quote of the Day.
- **Scan** Allows you to automatically ping across a range of IP addresses to find responding hosts. It also allows you to scan TCP ports to determine which ports are accepting connections; this indicates the services running on a server and often allows you to identify the operating system running on the host.
- **SNMP (an SNMP MIB browser)** Allows you to get low-level SNMP information from managed network devices and hosts.
- **WinNet (a NetBIOS probe)** Returns Windows Networking information about hosts on the local network.
- **About** Provides information about your local host's TCP/IP configuration.

The Scan tab is the most useful for hackers, as it identifies targets of opportunity within a specific IP address range. You can use port scanning to see quite clearly what your firewall vulnerabilities are from the Internet, and therefore protect yourself in advance. Figure 13.6 shows the results of a scan against a wide variety of computers.

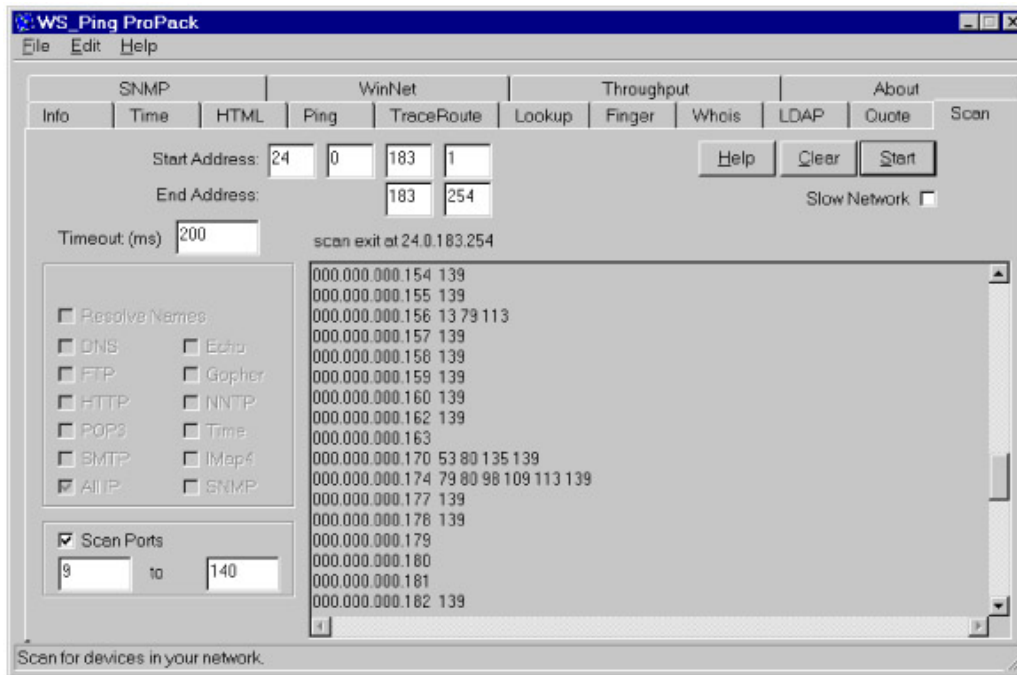


Figure 13.6: Results of a scan against a network
Internet Scanner

Internet Security System's Internet Scanner is the most comprehensive security checking utility we've found for Windows, Linux, and Solaris. It checks a vast array of common security problems and ranks them according to the level of risk they present. The security scanner is client/server based, so you can scan systems remotely. Figure 13.7 shows the ISS Internet Scanner in action.

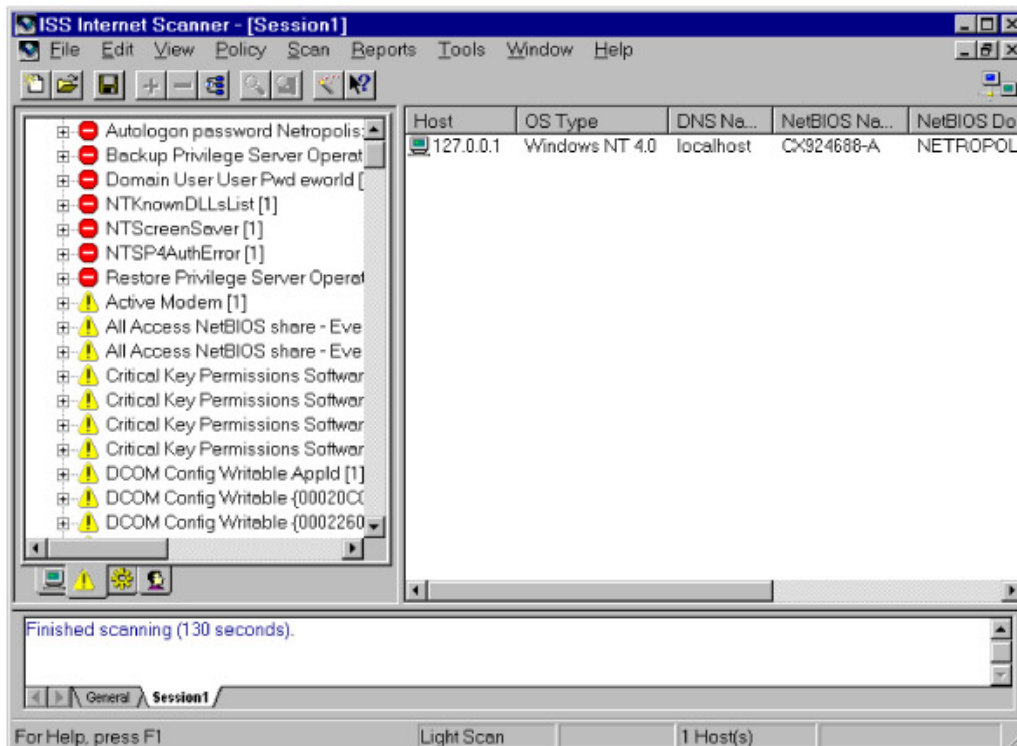


Figure 13.7: The ISS Internet Scanner

Installing IIS Internet Scanner is straightforward, but you will have to install a raw packet driver manually and reboot your system. Once installed, you simply launch the application and select

whether you want a light, medium, or heavy scan. These various grades indicate the depth (and time) Internet Scanner will devote to uncovering security problems in your system.

When finished, you'll see each problem listed in a display window. The Internet scanner can generate a report in HTML or text format that delineates each problem found and its effects. This report is very useful in determining what measures you'll take (if any) to correct the problems found.

A number of the problems turned up by the scanner (especially in the low risk category) can't be avoided because they are inherent risks in providing service to clients. You should concern yourself mostly with correcting those problems shown as high or medium risk.

Warning Never rely too heavily upon a single tool for security administration. Tools cannot check for problems their creators weren't aware of, so they go out of date quickly. Update security analyzers often.

Protocol Analyzers

Protocol analyzers are the test equipment of networks. They receive and decode the low-level packet information for every frame that travels across the link they're attached to. Protocol analyzers are heavily used by hackers to ferret out network information from connections, look for bugs or vulnerabilities in protocols, and sniff for passwords or other improperly encrypted high-value data.

Sniffer Basic (Formerly NetXRay)

Sniffer Basic is a protocol analyzer (commonly known as a packet sniffer) that runs under Windows and provides a very user-friendly graphical interface. Like all modern packet sniffers, Sniffer Basic provides features for packet capture and decoding. But unlike many others, it can provide graphical charts pinpointing exactly where in your network heavy traffic congestion is occurring.

Sniffer Basic shows Data Link layer frames and the data contained within them. The top window is the frame buffer, which shows each captured frame. Selecting a frame in the frame buffer allows you to view the frame's contents in the two content viewers below. The content viewers show you the raw (bottom) data and the decoded (top) information represented by that data. Reading from top to bottom in the Decode button is like reading up the OSI stack, from the Data Link layer (Ethernet) to the Network layer (IP) to the Transport layer (UDP) and finally the Session layer (DNS). Since this packet doesn't actually carry user data, it stops at the Session layer. The user interface for Sniffer Basic is very similar to the interface for the Windows Network Monitor.

Sniffer Basic is far too complex to detail its use here, but we can browse through a list of its major features:

- Client/server architecture allows you to attach to other copies on other machines for remote monitoring.
- Triggers alert you to conditions that you specify based on errors, utilization, or any other network characteristic.
- Packet Generator allows you to put your network under very specific load so you can test its ability to deal with heavy conditions. You can also use the packet generator to transmitted custom (or forged) packets.

Microsoft Network Monitor

The Microsoft Network Monitor is the protocol analyzer included with Systems Management Server (SMS), which is a BackOffice network administration utility. Network Monitor is the same software that ships with Windows as the Network Monitor, but it includes a special promiscuous mode NDIS driver that allows the software to monitor all the traffic on the connected network, not just traffic going to or from the server. This makes the software generally equivalent to robust protocol analyzers like Sniffer.

CommView

CommView is a lightweight, low-cost network monitor for Windows. Weighing in at about \$100 and very comparable to high-end sniffing tools, it's easy to acquire even for the most budget-constrained network administrator. You give up robustness; CommView can easily be overwhelmed trying to sniff all the packets on a 100MB network, so be sure to have good filters in place before you attempt to sniff a high-speed stream. CommView is perfectly fine on lower speed (10MB and below) circuits, which covers the vast majority of permanent Internet connections.

CommView is very easy to use, and has a very cool "Concurrent Connection Status" page that shows in real time how the connections on your segment are being used. The interface is very easy to use and makes it simple to jump back and forth within a TCP packet stream. CommView also has a very cool TCP reconstruct function where it can reconstruct a data stream by joining packets in order and stripping out the header information. The resulting text is the stream originally sent between the hosts and can be inspected as ASCII text, as HTML (if it's a web stream), or as a hex dump.

CommView supports the normal range of filtering operations and can filter on basically any header information. Each packet is deconstructed into Ethernet, IP and TCP/UDP, but CommView does not include modules for protocols above the Session layer. CommView can be downloaded from <http://www.tamos.com/>.

TCPDump, IPTraf, and Snarf

There are of course sniffer and packet logger utilities for Unix, but there you trade off the convenience of an easily installed commercial software package for the reduced expense of free software. (Because time is money, don't be under the impression that free software costs nothing—all these packages take some sophisticated configuration.) Three packages in common use are TCPDump, IPTraf, and Snarf.

TCPDump is a tool for Unix that records all of the IP headers for the network traffic that it can view. TCPDump puts the adapter in promiscuous mode, and if the adapter is connected to an unswitched Ethernet setup, it can record the headers of all of the traffic on the broadcast domain it is a part of. The log files are a little cryptic, but they contain all the information you need to identify network attacks. Note the ICMP redirect below:

```
11:15:52.431315 192.168.1.103.netbios-ns > 192.168.1.255.netbios-ns:
>>> NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST

11:15:52.501315 proxy1.rdc1.sdca.home.com.domain >
192.168.1.101.32770: 27437 NXDomain[|domain]

11:15:53.021315 192.168.1.103.1039 > 63.241.3.94.http: S
```

```

2465090815:2465090815(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

11:15:53.021315 192.168.1.101 > 192.168.1.103: icmp: redirect
63.241.3.94 to host 192.168.1.1 (DF) [tos 0xc0]

11:15:53.021315 192.168.1.103.1039 > 63.241.3.94.http: S
2465090815:2465090815(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)

```

That's a little hard to read. IPTraf does better, as you will see next. In addition, IPTraf will display and log Ethernet frames and record traffic based on MAC addresses. In this example, the TCP redirect attack is a little more obvious.

```

Mon Oct 22 11:20:45 2001; ***** IP traffic monitor started *****

Mon Oct 22 11:20:52 2001; TCP; eth0; 48 bytes; from 192.168.1.103:1041 to
63.241.3.94:80; first packet (SYN)

Mon Oct 22 11:20:52 2001; ICMP; eth0; 76 bytes; from 192.168.1.101 to
192.168.1.103; redirect

Mon Oct 22 11:20:52 2001; TCP; eth0; 48 bytes; from 63.241.3.94:80 to
192.168.1.103:1041; first packet (SYN)

Mon Oct 22 11:20:52 2001; ICMP; eth0; 68 bytes; from 192.168.1.101 to
192.168.1.103; redirect

```

Snarf is a tool that can do more than just log headers. It can copy the entire contents of packets to a log directory on your computer and it can also generate alerts on certain occurrences. The snarf alert file for the following traffic is shown here:

```

[**] [1:472:1] ICMP redirect host [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/22-11:38:30.301315 192.168.1.101 -> 192.168.1.103
ICMP TTL:255 TOS:0xc0 ID:12800 IpLen:20 DgmLen:355 DF
Type:5 Code:1 REDIRECT
[Xref => http://www.whitehats.com/info/IDS135]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0265

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 192.168.1.101 (THRESHOLD 4
connections exceeded in 1 seconds) [**]
10/22-11:39:45.237651

[**] [100:2:1] spp_portscan: portscan status from 192.168.1.101: 5 connections
across 5 hosts: TCP(4), UDP(1) [**]

10/22-11:39:59.017275

```

Encryption Tools

Encryption tools help you store and transmit your data securely. Two varieties are covered in this section: e-mail cryptography and disk encryption.

Transparent Cryptographic File System

TCFS is an open-source file-level encryption system for Linux based on NFS that is very transparent to clients. The encryption and decryption of files take place on the client machine, not the server. This puts the compute load on the client, ensures that only encrypted data travels over

the network, and ensures that encryption keys are stored only on the client; they do not travel over the network. TCFS is available at <http://tcfs.dia.unisa.it/>.

Encrypting File System (EFS)

Windows 2000 and later versions support built-in file encryption via the Encrypting File Service (EFS). Encrypting File System is a file system driver that provides the ability to encrypt and decrypt files on the fly. The service is very simple to use: users need only check the encrypted attribute on a file or directory to cause the EFS service to generate an encryption certificate in the background and use it to encrypt the affected files. When those files are requested from the NTFS file system driver, the EFS service automatically decrypts the file for delivery.

Why Nobody Actually Uses EFS

Although file encryption sounds like a really cool feature, the current implementation in Windows 2000 is so flawed that EFS is mostly worthless, except perhaps on laptops.

The biggest disappointment with EFS is that it only works for individual users. That fact alone makes it useful only on client computers. Encryption certificates for files are created based on a user identity, so encrypted files can only be used by the account that created them. This is extremely shortsighted on Microsoft's part. If encryption certificates could be assigned to group objects rather than just accounts, encryption could be used to protect general files stored on a server.

Of course distributing certificates to members of the group would require security, but Kerberos is capable of creating session keys that could be used to encrypt certificates to keep them secure while they're being transported to members of a group.

EFS will not allow encrypted files to be shared because it decrypts them before it delivers them. This too is shortsighted. If encryption certificates were held by a group, the encrypted file could be delivered over the network to the client in its encrypted state, and the client computer could use its participation in the group having the decryption certificate to decrypt the file. This facility along with Kerberos could keep file sharing secure enough to use over the Internet without a private tunnel.

Furthermore, the Achilles' heel of encryption (loss of the encryption certificate) would not be such a big deal. As long as any user who was a member of the group having the certificate still existed, that user would still have a copy of the certificate that could be copied to other users or used to decrypt the files.

Alas, EFS (like file replication) is another example of a service that would have been really cool if Microsoft had done it right. As it is, they did just enough to say they had file system encryption.

Aside from the fact that it only works for individuals, there are a number of other problems with EFS.

Encryption certificates are stored by default in the registry of the local machine, where they can be easily recovered and used to decrypt the files on the machine. For EFS to function correctly as a secure encryption service, certificates must be moved from the local machine to a physically secure certificate server or exported to removable media that isn't left with the machine. Since the vast majority of users don't know or do this, the file system should be more accurately called the Obfuscating File Service.

Using SysKey (Start ➤ Run ➤ Syskey) to move the System Key (a hash used to encrypt the local

SAM accounts database which contains the EFS decryption certificate) to a floppy or to be entered as a password at boot time is the only way to secure local EFS certificates—and that password or floppy must be available to everyone who needs to boot the computer.

Performing drag-and-drop copy operations into an encrypted folder will not automatically encrypt the file, because Move operations do not change the attributes of a file. Cut and Paste operations do, because you're effectively deleting the old file and creating a new one. This is a major problem with the Explorer interface because most people don't know that fact and because most people move files using drag-and-drop operations. This means that many EFS users believe their files are secure when in fact they're stored as plaintext files.

Encrypted files will be decrypted if they are moved to a non-NTFS volume. This is because non-NTFS file systems do not support encryption. Be certain you know what file system type you're copying encrypted files to. In my opinion, EFS should copy the file in its encrypted state to the non-NTFS store, where it would be useless until it was moved back to an NTFS store. No crypto service should automatically decrypt anything without the consent of the user.

Encrypted files cannot be shared. This restriction is designed to keep them encrypted—shared files are sent as plaintext over the wire, so anyone with a network sniffer could decrypt them. A more robust encrypting file system would simply transmit the encrypted file over the wire and trust that the receiver had the appropriate certificate to decrypt the file.

Many programs (most Microsoft Office programs) create temporary files either in the local directory or in the temp directory when a file is being edited. Encryption for temp files follows the normal rule: If the file is created in a folder that has the encryption flag set, the temp file will be encrypted. Otherwise, it will not be and would act as a foil to your encryption security. A solution to this problem is simply to set the encrypt flag on the temp directory to ensure that all temporary files are encrypted. You also need to be sure you know exactly where temp files are being created—just because the temp directory exists doesn't mean that every program uses it. Many third-party applications create their own temp directories.

Another vector for accidental decryption is printing: when you print a document that is encrypted, the file is decrypted by the source application and sent via plaintext to the print spooler. If the spooler is configured to spool documents (as most are), then the printed data will be written to a file that could be undeleted to recover your encrypted data. The solution to this problem is to either configure printing to avoid spooling documents throughout your network or to set the encryption flag on the folder that the spooler uses to create spooled documents.

Tip EFS requires a Data Recovery Agents policy, so you can disable EFS throughout a domain by creating group policy object for the domain that specifies an empty policy for encrypted data recovery agents. The policy must exist and it must be empty, because a no policy setting will allow the local workstation's policy to override the domain policy.

I don't recommend using EFS except on single user computers that can't otherwise be physically secured. It's easy to use, but its ease of use is a security blanket rather than real security. There are so many holes in EFS and so many accidental decryption vectors that it would be hard for a security expert to remember them all, much less a typical user. EFS is valuable for high theft computers like laptops that are configured to encrypt the print spool directory, temp folders, and My Documents. For everyone else, it's worse than nothing because it makes you feel secure when you probably aren't.

PGP

Pretty Good Privacy (PGP) is strong e-mail and disk encryption software written by Phil Zimmerman and released for free to provide strong encryption for everyone. Phil's company (also called PGP) has merged with Network Associates (NAI), but a freeware version of PGP is still available at <http://www.nai.com/>. PGP will automatically install support for file encryption and e-mail encryption on Win32 platforms, and has plug-ins for Eudora, Outlook, and Outlook express on Win32 platforms. It also has strong Macintosh support, and many variations for Unix.

PGP provides the strongest cryptography that is publicly available and uses your choice of algorithms with key lengths up to 4096 bits. (Every bit takes twice as long to crack; compare that to the standard 128-bit security to get a feel for how much stronger PGP is than standard cryptography.) Perhaps the coolest thing about PGP is the fact that NAI and MIT maintain public key servers upon which you can store your public key for others to encrypt messages to you when they don't already have your key. That service makes PGP highly useful, and it is the reason why it's the ubiquitous encryption solution.

Scramdisk

Scramdisk is my favorite disk encryption tool for Windows. It's very inexpensive (\$20/ user), robust, and works with all versions of Windows. It includes a very cool steganographic feature that allows you to embed encrypted data inside a WAV sound file (that remains playable even with encrypted data stored inside it).

Scramdisk creates encrypted disk files by generating its random keys using keyboard and mouse input. You assign four passwords to each encrypted file in order to create a long and strong secret key for unlocking it.

Scramdisk runs as a Tray Icon that can be used to control mounting and dismounting encrypted disk files. Encrypted disk files appear as normal drives to the system, so once they're mounted you can use them just like you'd use any drive. Scramdisk includes support for all the popular encryption methods: Blowfish, 3DES, DES, etc.

You can download Scramdisk from <http://www.scramdisk.com/>.

Tip My company uses Scramdisk along with USB flash disk keychain dongles to store secure passwords and keying material for VPNs. By physically keeping this information on our keys, we ensure that our laptops (which have drivers to mount the keychain dongles and Scramdisk to decrypt them) and keys won't be simultaneously lost in the same place.

Thawte Certificates

Thawte Consulting (<http://www.thawte.com/>) is the second largest root certification authority in the world, and unlike all other root-CA's, they don't charge for personal encryption certificates to encrypt individual e-mail. This allows you to use compatible e-mail software like Microsoft Outlook or Netscape's integrated e-mail software to automatically transmit your public key to others, add their keys to an encryption keyring, encrypt mail using the recipient's key (if you have it), and decrypt your encrypted mail. Best of all, the encryption and decryption processes are nearly transparent (Outlook does feel compelled to issue lots of modal dialog boxes like "Encrypting Message Using Public Key!" that are annoying). Get your free certificate today!

Password Strength Checkers

Password strength checkers read the encrypted passwords stored on your server (when run locally) and run brute-force decryption against them. The longer it takes to decrypt the passwords, the stronger they are. Network-based strength checkers perform remote logon attempts automatically at a high rate, but are far slower than locally executed password strength checkers.

L0phtCrack

L0phtCrack is a password strength checking utility that will allow administrators (and hackers, on improperly secured systems) to view the passwords that users select. L0phtCrack was developed by a hacking group to expose the security risks inherent in Windows NT.

By providing backward compatibility with LANManager clients, Microsoft has drastically reduced the password strength of Windows unless an administrator specifically disables LANManager compatibility. Valid LANManager passwords are far less secure because they allow a restricted set of characters and because they are limited to 14 characters in length. In addition, they are grouped into two 7-character passwords, which makes computing the password far easier, because it takes large orders less computing power to decrypt two 7-character passwords than one 14-character password. Figure 13.8 shows passwords being decrypted on a Windows NT Server.

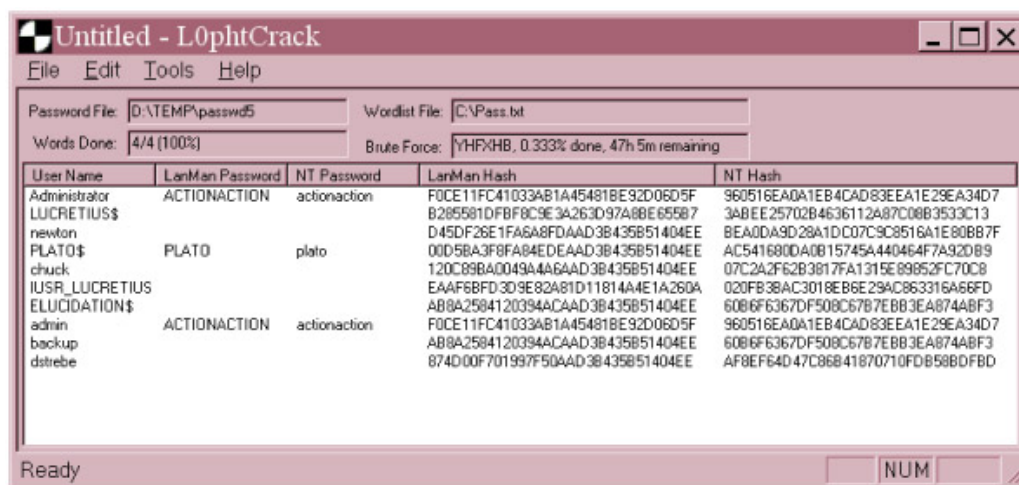


Figure 13.8: L0phtCrack decrypts passwords on an NT 4 Server.

An Administrator can run L0phtCrack on a server to dump password hashes directly from the registry and then run a brute-force decryption to expose the plain text passwords chosen by users. This allows you to determine if users are using strong passwords or passwords that can be easily deciphered.

Warning Just because you've disabled LANManager authentication doesn't mean your passwords can't be brute-forced by crack tools like NT Crack. It just means that it will take (a lot) longer.

NetBIOS Auditing Tool

The NetBIOS Auditing Tool is technically a NetBIOS security auditing tool. Its purpose is to expose security flaws in NetBIOS networks, but there are tools better suited to that task.

The NetBIOS Auditing Tool's automated password checking feature makes it a favorite of hackers. Using the NetBIOS Auditing Tool, a hacker can launch an automated attack against an NT Server

from over a network and attempt to attach via NetBIOS (which allows the remote user to map a drive, for instance) by repeatedly trying passwords against accounts. The NetBIOS Auditing Tool accepts a list of accounts and passwords and then tries each password against each account in the order presented.

The NetBIOS Auditing Tool accepts three command line parameters:

```
usage: nat [-o filename] [-u userlist] [-p passlist] <address>
```

- `-o` specifies a log file to which output redirects.
- `-u` specifies a text file of account names, one per line, to attack.
- `-p` specifies a text file of passwords to attempt against each account.

A few normal security measures can prevent NetBIOS Auditing Tool attacks. If user accounts are to time out after a few password attempts, these sorts of hacks will never work against them. But the NT Administrators account cannot be locked out. Since this is the account hackers are after, the NetBIOS Auditing Tool is still an excellent tool.

Foiling NetBIOS Auditing Tool attacks can be accomplished by renaming the administrator account and by restricting administrator account logons from the network. You can then create another administrative account (that can be locked out) and use that for network administration.

Tip You can create a simple intrusion detector by renaming the administrative account and then creating a normal user account called Administrator that has no permissions to access anything and a complex password. Then, establish auditing on logon attempts for that account. Any time anyone tries to use it, you are under attack.

Personal Firewalls

Personal firewalls are lightweight software firewalls intended to protect personal computers and workstations from hacking attempts when the computers are attached directly to the Internet and do not have an intervening "real" firewall to protect them.

Because remote computers that attach through your VPN create an unsecured back door into your corporate firewall, you have to extend your security perimeter to encompass these devices. They must be firewalled, or you might as well not bother firewalling your network.

Placing an actual firewall device at the home of every user is of course ideal, but inordinately costly in most circumstances. NAT devices are less expensive and more convenient, but moderately less secure. When neither is practical (for instance, with laptop users who will be connecting from many different places), personal firewall software is necessary.

Personal firewall software works by "wedging" between the Ethernet adapter driver and the TCP/IP stack and inspecting communications going between the adapter driver and the stack. Some are multiprotocol, while most are effective only against IP (which is the only real threat anyway). Personal firewall software is typically easy to install and largely configuration-free; it makes most of the right decisions for users automatically, with varying degrees of effectiveness.

The biggest problem with personal firewalls is the instability that poor-quality offerings cause and the inconvenience that they place on the user. There are about twenty personal firewalls on the market for Windows. While the vast majority of them will secure your computer, there are

tremendous variations in quality, stability, usability, and convenience among them. Among the major offerings, very few rise to the level of quality and convenience that I would consider using or inflicting on the users of the networks I support. Our company uses Symantec Personal Firewall 2002.

Personal firewalls typically perform connection-oriented stateful packet filtering, outbound connection blocking, and HTTP content blocking that can strip out ActiveX, Java, and cookies (in premium editions). Because personal firewalls actually run on the computers, they have a security venue available to them that is missing on separated firewalls: They can block in or outbound connection attempts based on the process (application) that is attempting access, rather than simply relying on Network layer information. This allows them to block or control connections for applications like Windows Media Player, Real Audio, and so forth.

BlackICE Defender

BlackICE Defender was one of the early personal firewall programs on the market, and it basically defined the characteristics of those that followed. Unfortunately, its makers have not been vigorous about keeping pace with the market, and BlackICE Defender is getting a little long in the tooth. Most of the comparable programs in this section do a better job of securing your computer and are more up-to-date on the current threats—even the free ones.

BlackICE Defender lacks the "configure-as-you-go" policy editing featured in ZoneAlarm and Norton Personal Firewall. For most users, that methodology is ideal. In BlackICE Defender, you set a simple "Caution Level According to Four Settings: Trusting, Cautious, Nervous, and Paranoid," settings which map to various levels of inbound connection blocking.

BlackICE Defender lacks any outbound connection blocking, which I believe is crucial for preventing trusting users from accidentally compromising their systems, stopping the spread of NIMDA-like Internet worms, and for controlling privacy. It also lacks the ability to perform cookie, Active-X, and Java blocking in HTTP Streams.

While BlackICE Defender does catch and report blatant attacks like port scans, it allows standard Pings even on the highest security setting, which means that hackers running a ping scan will find your computer. This is a serious security failure in my opinion.

On the default security setting, I was able to run a NetBIOS information and automated password attack against BlackICE Defender without so much as a warning, as well.

In sum, BlackICE Defender is easy to install and use, but less secure and not as easily customizable as its comparably priced competitors. The death knell in my opinion is that the free version of ZoneAlarm is better.

Norton Personal Firewall 2002

Compared to most of the personal firewalls reviewed here, Norton Firewall is a joy to use, simple to configure, and functions well.

Rather than rely on the user to configure security settings without really knowing what you want or need, Norton Personal Firewall presumes a medium security environment. To custom tailor your firewalling, Norton Personal Firewall asks a question each time a policy situation arises. You can elect to enable or disable the policy for that instance or in that situation from that point on. The net effect is that your computer is configured for decent default security, and then loosened up by the

user on a case-by-case basis. In my opinion, there's no better way to tailor a firewall policy.

McAfee Firewall 3.0

McAfee Firewall 3.0 is so badly flawed that we couldn't review it. When my coworker installed it on his laptop, he had to remove his wireless network adapter in order to attach the CD-ROM, so his primary network interface was not available during the software installation. As far as we can tell, this caused the software to disable DHCP, so that when he re-installed the network adapter, he couldn't get an IP address. We dug through the user interface and enabled DHCP—but to no effect. We couldn't get the interface to come up properly.

After trying to get the DHCP configuration to work, he attempted to disable the firewall service, which failed with an error message. When he decided to reboot, that didn't work either—the computer hanged at the shutdown screen dimming. We then forced a shutdown, and when the computer restarted, no network access could be achieved using any of the techniques we tried.

So he uninstalled the firewall software, also to no avail. The IP stack on his computer was so badly mangled that even with the firewall software removed, he couldn't use it.

So he uninstalled the TCP/IP stack from his machine in an attempt to correct the problem. This too failed to get him back on line. His next attempt was to remove every network adapter, service, and protocol in an attempt to completely remove the links to the McAfee Firewall service in the registry. In the end, he wound up restoring his computer from the original CDs to get back to work.

Like most of McAfee's software, this offering seems to be written to "feature match" the comparable offering from Symantec, but lacks the rigorous testing of boundary cases. Avoid the hassle by buying Symantec's offering.

CheckIt Firewall

CheckIt Firewall is the personal firewall offering from SmithMicro. Like nearly all of their software, CheckIt is functional but about a generation behind in terms of features and user interface, somewhat fragile, and not particularly well thought out. CheckIt lacks the "build-as-you-go" firewall policy paradigm and instead relies upon a wizard during setup to ask you questions about your security. Considering the price, there are much better personal firewalls available.

Note During the boot process for DHCP enabled computers, CheckIt will report a broadcast attempt without specifying that the broadcast is actually a DHCP plea. If you block it, you won't get an IP address.

Tiny Personal Firewall

Tiny Personal Firewall is free, lightweight, easy to use, and effective. It adds a bit of noticeable overhead in some situations, but it is certainly tolerable given the price and feature set. Tiny Personal Firewall is the Chevy to Symantec's Cadillac—conceptually they're very similar, but TPF is the no frills "just get me there" product. Of course, NPF is quite a bit bigger than TPF—43MB vs. 2MB installed.

The rule base generation system is very similar to NPF, as is the mode of operation. When network activity occurs, TPF prompts you whether to permit or deny access and has a selector for whether or not to make that rule permanent.

TPF lacks the HTTP content blocking of NPF. It won't stop cookies on a site-by-site basis (a feature I love for stopping pop-up ads, but which frankly doesn't have much to do with real security), doesn't stop ActiveX controls, and won't block Java.

For standard firewalling, it does an excellent job, and you can't beat the price if you're a non-commercial user. NPF is free for non-commercial users and \$20 for commercial users. If you're a business and have to pay, go with the similarly priced Norton Personal Firewall. You can download Tiny Personal Firewall from tinysoftware.com.

ZoneAlarm

ZoneAlarm is a free lightweight firewall for Windows computers. ZoneAlarm provides much the same interface and protection as Tiny Personal Firewall, with a bit more polished look to it. Their functionality is almost exactly equal. ZoneAlarm's licensing, like Tiny Personal Firewall's, makes it free for personal use, but requires payment for commercial use. Both products actually serve as introductory versions of more complete products offered by the installer for sale.

ZoneAlarm is easy to use and well behaved. It installs easily and builds its access policy the same intuitive way that Tiny Personal Firewall and Norton Personal Firewall do. ZoneAlarm's user interface is considerably nicer than Tiny Personal Firewall, and it represents the best of the available "free for non-commercial use" firewalls. It (or something better) should be installed on every Windows home computer on the planet. You can download ZoneAlarm at <http://www.zonealarm.com/>.

Case Study: Hacking @Home

While preparing some of the security tools for this chapter on my home machine (which is connected by cable modem through @Home's service), I naturally ran a port scan against the class C IP range that my computer is a member of. Nearly 70% of the available IP addresses responded, and about 80% of those machines had the Net-BIOS port (139) open, indicating that they were either Windows with file sharing turned on, Windows NT, or Linux computers running Samba.

Purely for the purpose of gathering statistics for this book, I attempted to map drives on the computers that were listed using a command like:

```
C:\>net use z: \\10.0.0.138\c
```

A shocking 30% of NetBIOS clients allowed this mapping (or the related c\$ mapping for NT machines) without providing a password (or in the case of NT, a blank password).

Of course, I disconnected the mapped drives after taking the statistics.

If you use a high-speed constantly connected service like xDSL or cable modems, you need to be especially alert to the possibility of intrusion because you don't have a firewall between your personal computer and the Internet. I suggest using a dedicated NAT firewall like the very popular Linksys DSL/Cable Router to protect your network. These NAT firewalls make very strong lightweight firewalls and provide you with the ability to connect numerous computers to a single connection. If you can't afford that, at least download the free version of ZoneAlarm and use it.

Chapter 14: Intrusion Detection

Overview

If someone broke into your network, how would you know? There wouldn't be any muddy footprints. If you had a strong firewall that had good logging capabilities, you might find evidence of an attack in your logs, but a smart hacker could even get around that.

To make the case for rigorous intrusion detection beyond that provided by firewalls and their logs, consider the case of a classic e-mail virus: A worker receives an e-mail from a coworker's home account saying that he's found a copy of a file that's been missing for a few months. The worker clicks on the executable attachment that says it's a zip file, which installs a Trojan horse that lies in wait until it detects a period of keyboard and mouse inactivity for long enough to assume that the worker isn't looking at the computer. The Trojan horse then opens a connection to a hacker's computer. Even if your firewall is designed to block outbound connections on unusual ports (the vast majority are not), nothing prevents the hacker from serving his attack software on a common port like 80 (HTTP). Your firewall will merely see what looks like an HTTP connection flowing out of the network to a web server, a type of connection it sees thousands of times a month.

This sort of attack will get right past even a strongly secured stateful inspection firewall like Firewall-1 or SonicWALL. Only proxy-based firewalls like Gauntlet and Symantec Enterprise Firewall can be relied upon to reject improper protocol data on standard ports.

Even in that case, a clever hacker will simply use a binary data port like FTP that can only be filtered for initial connection data; the true binary file data cannot be filtered because there's no way to predict what the file should contain. The hacker designs the Trojan horse and attack server to transmit fake session establishment data, while the client appears to be merely uploading a file, but is in fact uploading screen images and accepting mouse and keyboard input. A well-designed Trojan horse could even work through an FTP proxy. Any other binary protocol could also be exploited.

If you rely upon firewall logs to tell you when an intrusion has occurred, you'll never find this sort of attack because it will appear to the firewall as if it were a regular client-initiated FTP upload session. Nothing about it will set off any triggers or alarms.

So we've established that even the strongest firewalls cannot prevent certain attacks. Any useful connection to the Internet is a potential vector for attack.

This chapter covers how to secure your network against those attacks your firewall can't prevent, how to determine when you've been (or more importantly, when you're being) attacked, and how to assess the scope of the damage should an intrusion succeed. This chapter covers many intrusion detection techniques that you can use without spending additional money on specialized software, as well as some of the major software packages available for intrusion detection.

Direct Intrusion

This chapter is concerned primarily with detecting intrusion into your network from the Internet. But before we discuss TCP/IP and Application layer intrusion detection, it's important to understand that intrusion takes many forms at many other layers in your network. Direct intrusion, where someone gains physical access to your facility and sets the stage for further networked intrusion, is a rare but

important security problem that must be addressed to achieve holistic security.

Hackers are notoriously nonchalant, and have simply walked into businesses to get data directly or install software to propagate a further penetration into the network. If your company has secrets worth stealing, foreign espionage agencies are known to go to extraordinary lengths to acquire information in their national interest. Many foreign governments also ask their agents to acquire information in the economic interest of the country's large businesses.

The attacks in this section are exceedingly rare; most companies need not worry seriously about physical security. But if your company performs any research and development activity, then you should use more stringent security policy to protect the product of your research.

Real intrusion prevention begins with premises security, Physical layer security, and Data Link layer security. If your network is so fortified against Internet attack that a dedicated enemy cannot breach your defenses, they will change tactics and intrude more directly.

Possible vectors for attack include:

- Impersonating an employee
- Impersonating service personnel
- Wiretapping public data links
- Adding devices to the network
- Outright theft

Do you know everyone who works at your company? You don't unless you work at a small business. Does your company issue ID badges that everyone wears? They probably do not if you work at a small business. Employee impersonation is particularly risky, especially in medium-sized businesses—attacks of this sort are extremely rare.

Are new employees subjected to a background check? Although it as rare as any of the attacks in this section (and more frequently the subject of movies than reality), if your organization had secrets worth more than \$50,000 to steal, it becomes worth the effort for an intruder to simply be hired in order to gain access.

Impersonating service personnel is the easiest way to gain trusted access to a company. If a phone repairman walked in and told your receptionist or security guard that they were experiencing telephone problems in the building, would that receptionist or security guard call to verify their story or would they simply escort them to the wiring closet? Would they know the difference between the attachment of a legitimate bit error rate tester (BERT) to a T1 line and an illegitimate wireless bridge?

If a salesman showed up and offered and demonstrated a new laptop, and said his company would be willing to let your staff evaluate the device for a month at no charge, would you accept?

If you hired a security expert to evaluate your network, would you bother checking her credentials? I've won a number of contracts to evaluate network security based on my experience and the fact that I've written a number of security-related books—but I've never had a customer check my driver's license to see if I was actually who I said I was. For some reason, companies go to reasonable effort to check out employees, but they let contractors and consultants parade around the company without so much as a look at their personal identification.

If you fired an IT staffer, are you certain that he hasn't embedded a Trojan horse or opened a back

door somewhere? Did you change every password in every device that the staffer had access to? This attack is by far the most common of those discussed in this section, and by far the most damaging because the attacker has intimate knowledge of your architecture, methods, and weaknesses.

Any of these examples of lax facility security could lead to a network intrusion. A minute alone with a firewall is long enough to modify the policy to allow a surreptitious service port entrance for further exploits, or to change the policy for an existing service. The policy abstraction allowed by modern firewalls is nice, but nothing prevents a hacker from creating a service called SMTP on port 5900 that actually accepts VNC (remote control software) connections. All you'd see in your rule base is that SMTP allows inbound connections; you'd have to dig to find out that *that* SMTP wasn't SMTP at all.

Intrusion Tools and Techniques

Hackers use a variety of tools and techniques to attack networks. A typical intrusion takes the following form, assuming that the intruder begins with no information about your site other than its address—and lately, not even that. A constant barrage of address and port scans reveal hackers rummaging through the Internet looking for targets of opportunity. When our company recently installed a firewall on a newly provisioned, never-before-used IP address, it took only seven minutes to log and drop its first hostile port scan. Slashdot.org reports default IIS and Linux installations being compromised routinely within minutes of being exposed to the Internet without protection. You can no longer count on obscurity as any sort of security.

Hacking attempts usually proceed as follows:

1. IP address scans
2. Port scans
3. Services evaluation
4. Target selection
5. Vulnerability probes
6. Automated password attacks
7. Application-specific attacks

Each of these attacks is detailed in the following sections:

- **Address Scans** Scan across the network range, if any, to find service hosts. Hackers usually scan at least the entire range of IP addresses around your host and may use reverse DNS lookup to determine if those other hosts are registered to your company. For this reason, you should assume they'd find any public hosts you have on the Internet, even if you didn't publicize its address.
- **Port Scans** Scan across responding hosts to find running services. This information tells the hacker what services are running on each publicly reachable host. Port scans typically work through firewalls as long as a host can be reached, especially if the scan is limited to service ports like 21 and 80 rather than scanning across all ports (which some firewalls are capable of detecting immediately and blocking on).

Reality Check: Target or Opportunity?

Opportunistic hackers and automated worms searching for random targets don't bother with complete port scans; rather, they scan only for the port required by the specific exploit they

know how to perpetrate. For example, when a hacker has acquired code that can exploit an unpatched web server, they scan only port 80 in search of vulnerable servers. When the port is found, sophisticated exploits will probe for information to determine if the web server is of the correct type for the attack (typically, a simple HTTP page request will suffice)—simpler exploits just push the attack whether it will work or not. The exploit will then push its attack against the server to compromise it.

The important difference is this: If your firewall or IDS (Intrusion Detection System) reveals a complete port scan, then someone is specifically targeting your organization, and you've got a serious problem. If the log reveals a single port scanned, then a worm or opportunistic hacker is merely looking for a target of opportunity, and the problem is of little consequence as long as you're proactive about patching your public servers and using Application layer firewalls to eliminate service-specific attacks.

- **Services Evaluation** Determine the operating system type of each host. After probing common service ports like Echo, Chargen, FTP, Telnet, SMTP, DNS, HTTP, POP, NNTP, RPC locator service, NetBIOS, NFS, etc., the hacker will determine what operating system each host appears to be running. Windows-based hosts typically respond on NetBIOS ports but do not respond on Telnet, whereas Unix hosts respond on Telnet but not on the RPC Locator service used by Windows NT. Linux hosts in their default configurations respond on a wide array of services and are easy to spot for that reason. It's a simple matter for any one of a number of text responding services like Telnet, FTP, HTTP, SMTP, or POP to receive a service banner indicating which specific application and version is providing the service. Since most applications have an affinity for certain operating systems, determining the operating system is trivial.
- **Target Selection** Selects the weakest found host. Hackers will usually target the host with the most running services, in the assumption that little to no work has gone into securing that host's default configuration. Windows hosts that respond on port 139 (NetBIOS) are certain to be attacked, since exploiting that service can lead to full control of the machine. Other services, like Terminal Services, VNC, pcAnywhere, or other broad-spectrum services that provide remote control are popular targets for attack.
- **Service-Specific Probes** Uses vulnerability analysis tools like SATAN against Unix systems or the Internet Scanner from Internet Security Systems for Windows hosts. These probes check for a wide range of known service vulnerabilities that are easy to exploit, so they're checked first.
- **Automated Password Attacks** Used against services like FTP, HTTP, NetBIOS, VNC, or others that allow access to the file system or a remote console. Hackers employ software specifically written to perform a high rate of logon attempts (like the NetBIOS auditing tool) using dictionaries of common passwords. Failing this attack, most hackers will concede defeat or resort to simple denial-of-service attacks if they hold a grudge against you.

Warning VNC, the popular free remote control program, is especially susceptible to automated attacks. First, it typically installs on a unique and easily scanned address. Secondly, it is shielded only by a single password, not by a user account and password. Finally, all versions prior to 3.3r7 respond immediately to failed logins and do not lock out after numerous attempts. Hackers have created high-speed password crackers for VNC that can gain access to machines exposing the service in short order.

If a hacker ever gains console access to a machine, they're certain to run a high-speed local

automated password cracker like Crack or NT Crack against your host to exploit other accounts.

Hackers have also been known to set up seductive websites offering free utilities to browse for account names and passwords. They've got your IP address when you visit. If you enter an account name and password, the software can associate the account and the IP address—so they know where you are and what identification you're likely to use. Do you ever use the same password and account name you use at work on websites? Like Microsoft's TechNet? Or the thousands of support sites for network software? Most people do. I do. This makes it easier for hackers to access your preferred account name and password.

- **Service-Specific Attacks** Comprises the remaining range of attacks a hacker might employ, and include the unusual, uncommon, or difficult tactics hackers might use if they really wanted to exploit your Internet servers and no previous techniques had worked. These attacks include buffer overrun attacks, source-routed attacks, hijacking attempts, network sniffing for passwords, or seductive e-mail to install a Trojan horse. Most of these attacks (except buffer overrun attacks) are exceptionally rare.

Hackers employ a wide body of software tools in their trade. Tools meant for administrators, like the SATAN and the Internet Security Scanner, become potent weapons in the hands of a hacker.

Hackers also exploit the specific software tools you use in your network. For example, enterprise firewalls have remote management applications, most of which are based on a fairly short shared secret password. Many firewalls have "hidden rules" that allow the attachment of their remote management client software in the mistaken perception that you'll always want to be able to remotely manage your firewall. Nearly every software firewall this book covers can be downloaded in a demonstration version for free from the Net. While the firewall engine might time out after 60 days, the management interface works forever. This means that every hacker on the planet has the remote tools to manage your firewall—all they need is your password.

Intrusion Detection Systems

Intrusion detection systems (IDS), also known as intrusion detectors, are software systems that detect intrusions to your network based on a number of telltale signs. Active response systems attempt to either block attacks, respond with countermeasures, or at least alert administrators while the attack progresses. Passive IDS systems merely log the intrusion or create audit trails that are apparent after the attack has succeeded.

While passive systems may seem lackluster and somewhat useless, there are a number of intrusion indicators that are only apparent after an intrusion has taken place. For example, if a disgruntled network administrator for your network decided to attack, he'd have all the keys and passwords necessary to log right in. No active response system would alert on anything. Passive IDS systems can still detect the changes that administrator makes to system files, deletions, or whatever mischief has been caused.

Inspection-Based Intrusion Detectors

Inspection-based intrusion detectors are the most common type. These intrusion detectors observe the activity on a host or network and make judgments about whether an intrusion is occurring or has occurred, based either on programmed rules or on historical indications of normal use. The intrusion detectors built into firewalls and operating systems, as well as most commercially available

independent intrusion detectors, are inspection based.

Intrusion detectors rely upon indications of inappropriate use. These indicators include:

- Network traffic, like ICMP scans, port scans, or attachment to unauthorized ports.
- Resource utilization, such as CPU, RAM, or Network I/O surges at unexpected times. This can indicate an automated attack against the network.
- File activity, including newly created files, modifications to system files, changes to user files, or modification of user accounts or security permissions.

Intrusion detectors monitor various combinations of those telltale signs and create log entries. The body of these log entries is called an audit trail, which consists of the sum of observed parameters for a given access object like a user account or a source IP address. Intrusion detection systems can monitor the audit trails to determine when intrusions occur.

Intrusion detection systems include these variations:

- **Rule Based** Intrusion detectors that detect intrusion based on sequences of user activities (called rules) that are known to indicate intrusion attempts, such as port scans, system file modifications, or connections to certain ports. The majority of intrusion detection systems are rule based. Rule-based intrusion detection systems cannot detect intrusions outside the realm of their programmed rules and are therefore usually ineffective against new types of attacks until they've been updated.
- **Statistical** Intrusion detectors that detect intrusion by comparing the existing base of valid audit trails to each new audit trail. Audit trails that differ substantially from the norm are flagged as probable intrusion attempts. Systems like these have the potential to detect hitherto unknown intrusion methods, but may miss rather obvious intrusions that might appear to be normal usage.
- **Hybrid** Intrusion detection systems that provide the best of both worlds by combining statistical and rule-based detection systems. Some of these systems are capable of creating new permanent rules from detected intrusions to prevent the intrusion from happening again without the overhead of statistical analysis.

IDS systems always require system resources to operate. Network IDS systems usually run on firewalls or dedicated computers; this usually isn't a problem because resources are available. Host-based IDS systems designed to protect servers can be a serious impediment, however.

Rule-based IDS systems can only detect known intrusion vectors, so all possible intrusions cannot be detected. Statistical intrusion detectors stand a better chance of detecting unknown intrusion vectors, but they cannot be proven to detect them until after the fact.

Because of these limitations, IDS systems generally require monitoring by human security administrators to be effective. Countermeasure technology and response systems that temporarily increase the host's security posture during attacks are all in the theoretical research stage. Current IDS systems rely upon alerting human administrators to the presence of an attack, which makes human administrators an active part of the intrusion detection system.

Decoy Intrusion Detectors

Decoy intrusion detectors, also called honeypots, operate by mimicking the expressive behavior of a target system, but rather than providing an intrusion vector for the attacker, they alarm on any use at all. Decoys look just like a real target that hasn't been properly secured. Because the decoy is not

normally used by anyone within your organization for any legitimate purpose, any connection to it at all is an intrusion attempt.

When hackers attack a network, they perform a fairly methodical series of well-known attacks like address range scans and port scans to determine which hosts are available and which services those hosts provide. By providing decoy hosts or services, you can seduce the hacker into attacking a host or service that isn't important to you and which is designed to alert on any use at all.

Decoys may operate as a single decoy service on an operative host, a range of decoy services on an operative host, a decoy host, or an entire decoy network. Decoy networks are very rare. Most decoy software runs on an operative host.

You can establish an effective decoy host by installing a real running copy of the operating system of your choice on a computer with all normal services active. Using your firewall's Network Address Translation, send all access to your public domain name to the decoy machine by default. Then add rules to move specific ports to your other service computers; for example, translate port 80 only to your actual web server.

When a hacker scans your site, he'll see all the services provided by your decoy host plus the services you actually provide on your Internet servers, as if they all came from the same machine. Because the services running on the decoy host include services that are easy to attack, like the NetBIOS or NFS ports, the hacker will be immediately attracted to them. You can then set up alerts to alarm on any access to those services using the operating system's built-in tools. You'll be secure in the knowledge that if the hacker intrudes into the system, he'll be on a system that contains no proprietary information. You can then let the attack progress to identify the methods the attacker uses to intrude into your system. I suggest installing a network monitor (like the one that comes with Windows NT) on the decoy host so you can keep logs of specific packet-based attacks as well.

Decoy hosts are highly secure because they shunt actual attacks away from your service hosts and to hosts that will satisfy the hacker's thirst for conquest, giving you plenty of time to respond to the attack. The hacker will be thrilled that he was able to break into a system, and will be completely unaware of the fact that he's not on your real Internet server until he browses around for a while. You might even consider creating a bogus "cleaned" copy of your website on the decoy server to maintain the illusion in the hacker's mind that the actual site has been penetrated. Any desecration performed on the decoy site won't show up on your actual site.

Best of all, decoy intrusion detection costs only as much as a copy of the operating system (NT Workstation can be used to decoy for NT Server, Linux can mimic any professional Unix server), target hardware, and your existing firewall. You won't have to pay for esoteric software.

Tricks of the Trade: Virtual Decoys

VMware is especially well suited for use as a decoy intrusion detector for lower-speed (<10Mbps) connections, because you can install a canonical decoy and then set the VMware virtual host disk mode to "non-persistent." VMware with a nonpersistent disk mode writes all sector changes to a temporary file that is destroyed each time the virtual machine is restarted. The net effect of this is that all changes made by a hacker are immediately lost when the virtual machine is rebooted, and your decoy host is completely restored. You're one click away from a completely fresh decoy no matter how badly defaced your decoy host becomes.

VMware also allows you to run multiple decoy hosts on a single machine. While you can manage

this with the Workstation version, you can set up an entire decoy network on the server versions of VMware, with numerous decoy hosts of all types to completely simulate an entire network for a hacker.

Finally, VMware comes with tools that allow you to perform low-level packet inspection on the network stream to virtual machines, which means that you can perform complete logging on the host that operates the virtual machines. That host, with its separate network identity and address, can be completely firewalled against compromise.

Available IDS Systems

Few reliable intrusion detection systems really exist. Firewalls with logging and alerting mechanisms are by far the most widely deployed, and the majority of those have no way to respond to an attack in any automated fashion.

Both Windows and Unix have strong logging and auditing features embedded in their file systems. Windows also has an exceptionally strong performance monitoring subsystem that can be used to generate real-time alerts to sudden increases in various activities. This allows you to create simple and effective IDS systems for your servers without adding much in the way of hardware.

Windows System

Windows has strong operating system support for reporting object use. This support manifests in the performance monitoring and auditing capabilities of the operating system, and in the fact that the file system can be updated with date-time stamps each time certain types of access occur. These capabilities make strong inherent security easy to perform.

File System and Security Auditing

The server versions of Windows have exceptionally strong support for file system and security auditing. You can configure Windows using Audit policies to create log entries in the security log each time any one of the following events succeeds or fails:

- Logon attempts
- File or object access, like copying or opening a file
- Use of special rights, like backing up the system
- User or group management activities like adding a user account
- Changes to the security policy
- System restart or shutdown
- Process tracking, like each time a certain program is run

What all this means is that you can create your own intrusion detection software simply by configuring Windows to audit any sort of behavior that could indicate an intrusion attempt.

Pervasive audit policy can slow down a Windows server dramatically, so you have to be careful of how wide-ranging your audits are in systems that are already under load. Audit on unusual events like use of user rights, logon and logoff, security policy changes, and restarts. Auditing on decoys is not an issue, however, because the decoy will only be responding to (hopefully) a few simultaneous hacking attempts. In any case, you're under no obligation to provide high-speed services to hackers—if it's slow, they'll simply assume your decoy is loaded as a production server would be.

File and object access is a special case in auditing. You have to enable file and object auditing and then use the Security tab of each file or folder's property panel to enable auditing for specific files. This allows you to limit the files that you audit. For system files, you should audit for writes, changes, and deletes. For proprietary or secret information you store, you should audit for read access.

File and object access occurs constantly, so if you audit a large number of commonly used files, you'll increase the amount of chaff (useless information) in your log files and slow down your computer. Audit only those files that are real intrusion targets, like the system files and your proprietary information.

There is a problem with Windows' audit policy: If a hacker actually gains administrative control of your system, the hacker is free to erase your security policy after it has been changed. To detect changes even in that event, see the next section.

Remember to configure your log settings to be appropriate for the purpose of the machine. On decoy machines, you want very large disk limits for log files, so that the machine can withstand numerous simultaneous sustained attacks. You probably shouldn't have it shut down when the log fills up, as that would leave the true server it's protecting up, and immediately reveal what's really happening to the hackers.

"Tripwire" for Windows

You can use the built-in functionality of Windows to test for changes to your system file in cases where you can't or don't want to use Windows' built-in file auditing system. The command prompt directory command can be used to display the last written-to time for a file by including the /TW (display last write time) switch, as in:

```
C:\>dir c:\winnt\*.* /TW
```

By redirecting the console output of that command to a file and storing that file on a removable media cartridge or over the network to another machine, you can compare it to the directory at a later date by reissuing the command and creating a new file. You can then use the file compare command-line utility to automatically compare the two files and point out changes between the initial write times of your system files and their current write times.

Many system files are written to frequently, while others never should be changed except after system updates or service pack installations. By recognizing which ones change routinely and which never change on your system, you can use this functionality to automatically detect unauthorized file system changes that have occurred on your system in much the way that Tripwire detects these changes in a Unix system.

In Windows, you should be particularly concerned about the following directories and their subdirectories (assuming your system drive is C: and that you've installed to \winnt; otherwise, replace the example with your system root):

```
C:\
C:\winnt
C:\winnt\system
C:\winnt\system32
```

To implement this system, type in the following batch file and use it to create your initial difference file for each protected machine:

```
@echo off

REM baseline.bat

REM Use this batch file to create a baseline

REM for file system changes.

Echo Creating Baseline...

Dir c:\*.* /TW >basel.txt

Dir c:\winnt\*.* /TW >base2.txt

Dir c:\winnt\system\*.* /S /TW >base3.txt

Dir c:\winnt\system32\*.* /S /TW >base4.txt

Echo Baseline created. Store baseline files

Echo In a secure location.
```

Whenever you suspect an intrusion, use the following batch file to create a comparison file that you can inspect with Notepad or any text editor:

```
@echo off

REM compare.bat

REM Use this batch file to create comparison

REM files for file system changes and to generate

REM the compared output

Echo Checking for system changes...

Dir c:\*.* /TW >comp1.txt

Dir c:\winnt\*.* /TW >comp2.txt

Dir c:\winnt\system\*.* /S /TW >comp3.txt

Dir c:\winnt\system32\*.* /S /TW >comp4.txt

FC basel.txt comp1.txt >root.txt

FC base2.txt comp2.txt >winnt.txt

FC base3.txt comp3.txt >system.txt

FC base4.txt comp4.txt >system32.txt

Del comp?.txt

Echo Finished finding changes. Changes are

Echo stored in the following files:
```

```

Echo root.txt stores changes to c:\

Echo winnt.txt stores changes to c:\winnt

Echo system.txt stores changes to c:\winnt\system

Echo system32.txt stores changes to c:\winnt\system32

```

Using this procedure, I deleted temp files in my winnt directory and added a comment line to the wintab.ini file. This resulted in the following output, which show the lines above and below each change as well as the change itself:

```

***** start.txt

01/05/99  01:06p                      7,023 WINNT32.LOG

03/03/99  07:21p                      1,799 WINTAB.INI

01/25/99  03:06a                      162,576 wjview.exe

***** FINAL.TXT

01/05/99  01:06p                      7,023 WINNT32.LOG

07/11/99  09:08p                      1,815 WINTAB.INI

01/25/99  03:06a                      162,576 wjview.exe

*****

***** start.txt

06/28/99  11:53p                      164 _delis32.ini

06/10/99  11:57a                      0 ~DF23BD.tmp

06/10/99  11:49a                      0 ~DF2544.tmp

06/10/99  11:43a                      0 ~DF29B9.tmp

07/01/99  09:32a                      0 ~DF2BF4.tmp

06/10/99  11:02a                      0 ~DF3240.tmp

07/11/99  06:17p                      0 ~DF3421.tmp

06/29/99  07:08a                      0 ~DF3B0E.tmp

06/23/99  11:07p                      0 ~DF3F97.tmp

06/24/99  06:44p                      0 ~DF4670.tmp

06/23/99  10:57p                      0 ~DF4B0E.tmp

11/05/98  06:33p          <DIR>          ~offfilt

***** FINAL.TXT

06/28/99  11:53p                      164 _delis32.ini

07/11/99  06:17p                      0 ~DF3421.tmp

```



```

11/05/98  06:33p          <DIR>          ~offfilt

*****

***** start.txt

06/11/99  04:45p          50 ?

          296 File(s)      119,295,337 bytes

                          1,031,680,000 bytes free

***** FINAL.TXT

06/11/99  04:45p          50 ?

          287 File(s)      119,295,353 bytes

                          1,031,700,480 bytes free

*****

```

Ignoring the first and last line of each block, the output clearly shows changes to the wintab.ini file's size and write time, the missing temporary files, and the difference in directory size.

Extremely clever hackers know how to change access times back to their original times and can be certain to make sure files are exactly the same size. So this method is not as theoretically pure as the method used by Tripwire, which relies on comparing cryptographic hashes made of each file. Hackers this careful are very rare though, so this method can be used to find the vast majority of file system changes.

You can also use the FC command to find changes in the system registry using a similar method that cannot be batch scripted:

1. Using the registry editor Regedit.exe, browse to the system key you wish to check for changes. You should be primarily concerned about the following keys:

```
HKLM\SYSTEM\CurrentControlSet
```

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion
```

2. With the key selected, select Registry ➤ Export Registry file.
3. Enter a filename on a removable media or network drive.

When you need to compare the files, repeat the process and use the following command to find changes to registry settings:

```
C:\>FC baseline.txt current.txt
```

The resulting text file will contain the changed registry settings.

Comparing file systems is sometimes the only method you can use to detect intrusion if the intruder has gained administrative access to your system in a manner that your intrusion detection systems can't track. If they've made any useful modification to your system at all, you'll be able to detect it with this procedure.

Unfortunately, you can't use this procedure to detect information that has simply been copied off your system. But Windows also maintains a "last read time" that can be output by the directory command, so if your security policy requires it, you can modify this procedure to look for changes in the last access time of any file. Be prepared to dig though—most system files are accessed very frequently. For that reason, you should define very narrowly the set of files you're interested in tracking and use this system to track only them.

Performance Monitor

The Windows performance monitor can be used to alert on sudden changes in resource utilization and on such attack indicators as a high number of logon attempts per second. The performance monitor can be configured to run programs to send e-mail, network alerts, or pages when any thresholds are reached.

Unfortunately the performance monitor is a user-level program that doesn't run as a service and only works while you are logged on. This makes it somewhat fragile. You can monitor machines remotely, however, so a single alerting workstation could be set up to monitor all your servers.

NAI CyberCop

Network Associates' CyberCop is a suite of four tools you can use for intrusion protection outside of simple firewalling. These tools are available for Windows and Unix, and can be downloaded for evaluation from NAI's website at <http://www.nai.com/>.

By integrating these tools with NAI's gauntlet firewall using their active security infrastructure, you can create a network security posture capable of automatically responding to threats and defending itself.

Tripwire

Tripwire scans files and directories on Unix systems to create a snapshot record of their size, date, and signature hash. If you suspect an intrusion in the future, Tripwire will rescan your server and report any changed files by comparing the file signatures to the stored record. Tripwire was an open-source project of Purdue University, but it continues development as a licensed package of Tripwire Security Systems, at <http://www.tripwiresecurity.com/>. The original open-source version is at <ftp://coast.cs.purdue.edu/pub/COAST/Tripwire>. A version for Windows is also available.

Case Study: Serious Intrusion

A friend of mine runs a network for a fairly large insurance firm, and they employ a number of sophisticated computers for both internal and external services. Being a cautious soul, he installed a number of intrusion detection and alerting systems on their public servers, but left the internal servers mostly alone, secure in the knowledge that their firewall would be able to block intruders from them.

One night at about 2:00 A.M., his pager went off and woke him up. He checked it; sure enough, it indicated an alert from the intrusion and service monitoring workstation that he used to monitor the servers and to collect intrusion data. Rather than indicating an intrusion attempt (his usual 2:00 A.M. wakeup), it indicated that one of the servers had gone down. Moments later, another page indicated that a second server had gone down.

He immediately booted his home computer and did a ping against his company's public web server. Sure enough, it gave no response. He assumed that a power outage had occurred that was long enough to cause the uninterruptible power supplies to force a shutdown. They would come back online as soon as power was restored, so there was no point in his traveling to work to sit in the dark and wait for that to happen when he could just as easily sleep.

Then it occurred to him that the monitoring workstation wasn't on the UPS because he considered it expendable. The monitoring station should have gone down first because it's in the same room and runs from the same source circuit as the rest of the servers, but had no battery backup.

This panicked him. It meant that some serious environmental problem, perhaps a broken water main or even an earthquake had caused some of his servers to fail. He immediately got dressed and drove the 20 minutes to work.

As he pulled up, he noticed a large van pull out of the parking structure and drive right past him through the empty parking lot. As it went by, he noted that it had no rear license plate mounted, but had the name of a common carpet cleaning company on the side.

He immediately called the police from his cellular phone and reported the van. When he entered the parking complex and went up the stairs, he saw that the outside doors had been forced open and could hear the alarm system inside the building going off. However, he'd had no response from the facility alarm people.

Fearing the worst, he entered the server room. The place was completely destroyed. Servers, hubs, routers, and switches had all been stolen out of their racks. Some computers lay broken on the floor, having been dropped in the vandals' haste. The vandals hadn't even bothered to unplug network cords; they pulled the power cords out and used a hedge trimmer to cut the masses of network cables going to the expensive switches and hubs they used.

The only computer that remained untouched was the fairly cheap workstation he used to perform ping testing and alerting for all the other servers. Even the firewall had been stolen.

He immediately called the alarm company to ask them why nobody had been called about the alarm intrusion. The alarm company operator told him that a number of alarms in that area had been set off that night, so they were checking for a system failure. Disgusted, he called his boss to explain the situation and then left to go back home to bed.

On his way back to the freeway, he saw two police cars ahead on the side of the road with the carpet cleaning van pulled over. He pulled over in front of the police cars, and when the officer approached, he went back with him to the van and identified his equipment as the vandals sat in the back seats of the police cruisers.

Amazingly, the van was stuffed with far more expensive equipment than they owned. Servers were stacked on top of each other like so many boxes of fruit. The police were surprised when my friend told them that although some of the equipment was his, it certainly hadn't all come from his company. After making his statement and going through the necessary procedures for actually getting his equipment returned (after it was entered into evidence), he went back home.

To make a long story short, it turned out that this van full of vandals had broken into nearly every company on that block, breaking open doors, going directly to equipment rooms, and then stealing network equipment. The numerous alarms that the alarm company had decided were spurious turned out to be real.

During the trial, in which my friend was a witness, it came out that one of the burglars worked for a janitorial service employed by the property management company that managed the entire block, which is how he'd identified which businesses had high-value equipment and where it was located. Another vandal worked at a used computer store that would have acted as a fence for the equipment.

Despite all the destruction and time lost, all the recovered equipment actually worked when they got it back. Apparently, the bandits had wrapped each computer in eggcrate foam before stacking them to protect their resale value.

Though the alarm company completely failed in their obligation to alert the company, my friend's intrusion detection software actually came through in a most unexpected (and literal) way.

Part IV: Operating Systems as Firewalls

Chapter List

Chapter 15: Windows as a Firewall

Chapter 16: Open Source Firewalls

Part Overview

Featuring:

- Firewalling features provided by Windows
- Firewalling features not provided by Windows
- How to configure Windows to protect itself
- Problems that Windows firewalling cannot solve
- Situations where Windows's firewalling support is appropriate
- Firewalling features of open-source operating systems like Linux and NetBSD
- Firewall software not included in standard distributions of open-source operating systems
- How to configure an open-source operating system to perform firewalling
- Problems with open-source firewall systems

Chapter 15: Windows as a Firewall

Overview

Since Windows NT 4, Windows has supported basic packet filtering to block ports and to help secure servers against attacks. Windows 2000 significantly improved the filtering to allow outbound filtering as well and to add support for Network Address Translation. Windows Server versions also support routing and PPTP VPN tunneling. Windows 2000 adds support for IPSec transport mode and for L2TP.

Despite the array of firewall features supported by Windows, the core TCP/IP stack is not sufficiently hardened to withstand serious attack from the Internet. Although many of the most egregious denial-of-service attacks have been patched against, any sustained flood attack will cause cascading failure that will at least render the server inoperable for a period of time.

Microsoft's Proxy Server products are also not hardened enough to compete as true firewalls. The new Internet Security Server appears to be much stronger, but is too new to have much real world information on.

This chapter covers the firewalling features of Windows, but you should always consider them to be supplemental protection for servers that are primarily defended by a true firewall.

Windows NT 4

The Windows NT 4 operating system is not a firewall. Windows NT supports simple packet and PPTP filtering, but not Network Address Translation or application proxy services without additional software. In addition, its TCP/IP stack is not hardened completely against malformed packets, although an NT 4 installation with all the latest patches will not fall pray to the most virulent attacks. Windows NT was not designed to operate as a firewall; rather, it was designed for higher network performance. The consistency checks that firewalls must perform on each received TCP/IP packet require a considerable compute load, which would be too much for a heavily loaded server to deal with. This is one reason why firewalls should be isolated on dedicated machines.

A Windows NT default installation's TCP/IP stack is not hardened and is vulnerable to a number of well-known exploits. For example, prior to the release of Service Pack 3, Windows NT did not check for the presence of a proper 0th packet in a fragment; rather, when a packet arrived with its end-of-fragment bit set, NT would simply conjoin the data it had already received, regardless of fragment numbering, and pass the data up. This meant that any hacker with a copy of Linux could build his TCP/IP stack to make every IP packet claim to be the 1st packet instead of the 0th packet. The packet could then pass right through most packet filters, which only check for TCP port information in the 0th packet of a fragment, and go straight to an NT machine. This exploit made simple packet-filtering firewalls pointless. It's one of the reasons that packet filtering alone does not constitute real firewalling.

Other exploits that can cause denial of service are various ping-of-death attacks. These attacks either send ping packets that contain more data than NT's packet buffers can deal with, or they deform the packet in a way that NT isn't prepared to deal with. For example, the attack could set the reply address to the address of the NT machine, which causes it to ping itself repeatedly at high speed.

Microsoft releases hotfixes on a regular basis to solve or deal with these hacking exploits, but you can count on at least a three-month turnaround between the development of an exploit and the release of a hotfix. Furthermore, many hotfixes are not fully tested, so even Microsoft does not recommend installing them unless you are subject to the problem they fix. Waiting for service packs that solve these problems can take up to a year. And so far, service packs have been hit-or-miss—Service Packs 2 and 4 actually caused numerous problems and were quickly supplanted.

That said, Windows NT does support some important firewalling functions natively without the addition of extra software. Although we do not recommend using Windows NT as a firewall to protect your network, these packet-filtering capabilities can be used to protect NT machines that must be exposed outside your firewall (Internet servers and PPTP end points). This means that you can put your Internet servers outside your firewall without having to put a firewall between them and the Internet. They won't be absolutely secure against attack, but they'll be in much better shape than they would be without these features.

Note Windows 2000 on the other hand supports Network Address Translation, which makes it a strong contender for the title of a true firewall. The IP stack is considerably more consistent than the IP stack provided with Windows NT 4 and can withstand a more concerted attack. (There's more on Windows 2000 later in this chapter.)

Capabilities

Windows NT supports three primary firewalling features:

- Packet filtering
- Encrypted tunneling
- Encrypted authentication

Unlike most modern firewalls, Windows NT cannot easily share firewall policy with other servers, but sophisticated NT administrators can create registry scripts that can be applied across a range of machines by clicking the script on each machine. Nonetheless, this minimal functionality makes it difficult to configure security consistently across a range of machines.

Windows NT's firewalling features are most appropriate in the role for which they were created: additional security on multipurpose servers. You can (and should) configure all your Windows NT servers to allow only those TCP protocols for services you intend to provide.

Packet Filtering

Windows NT provides a stateless packet filter. Stateless packet filters make their decisions based only on information contained within each packet; they do not retain information about connections or other higher-level constructions.

The packet filter is capable of blocking TCP, UDP, or IP protocols individually for each interface. The filter can only be configured to pass all protocols or to pass specific protocols. It cannot be configured to block specific protocols. The packet filter blocks only inbound packets. All outbound packets are transmitted.

Packet filters are configured by opening the network Control Panel to the TCP/IP protocol and clicking the Advanced button in the IP address panel. PPTP filtering, which blocks all packets except PPTP packets, can be enabled by checking the Enable PPTP filtering option.

Enabling all other forms of packet filtering is performed by checking the Enable Security option and then clicking the Configure button. You can then select the Allow Only radio button and enter the protocols you want to allow for each transport. Figure 15.1 shows the Windows NT packet filtering dialogs.

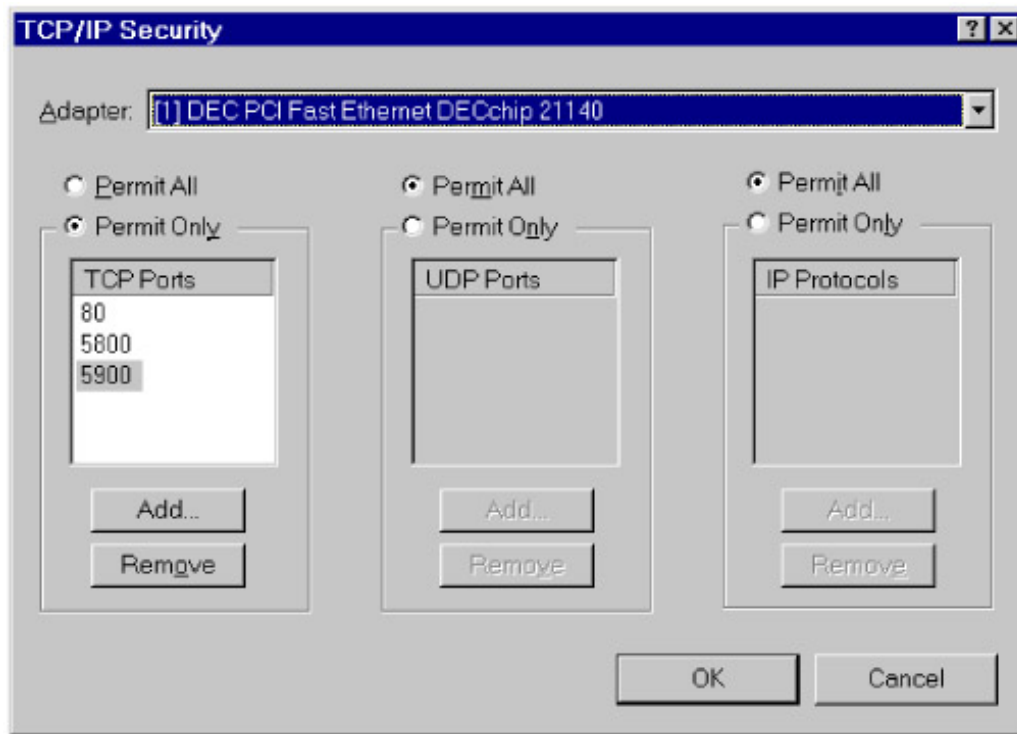


Figure 15.1: Windows NT Packet Filter Configuration

Tunneling

Microsoft included the Point-to-Point Tunneling Protocol (PPTP) with Windows NT to allow secure remote access. Microsoft provides two levels of security with PPTP corresponding to the U.S. government's previous limitations on export grade security and the security grade Microsoft considers to be ideal for encrypted communications. The 40-bit export grade security level is not particularly strong. The 128-bit domestic grade security package is strong enough for most uses. Unfortunately, it's not always clear which version you have installed, and the version will automatically change if you install certain service updates or use the wrong version of a service pack. You'll also have to make sure that both the server and all the clients support 128-bit encryption, and that your servers are configured to reject connections to 40-bit clients. Careful administration is the only way to be sure you're using the 128-bit domestic grade version.

Note The U.S. Government has since relaxed its restrictions on export grade security; Microsoft is allowed to export the 128-bit security and has made the upgrade available internationally.

In order to use the Windows NT Remote Access Service to create a VPN, you will, of course, need a Windows NT Server computer to host the RAS software. The computer must have sufficient RAM, hard drive space, and processing power to run Windows NT Server 4 adequately. (This means 64MB of RAM, a 4GB drive, and a Pentium or higher microprocessor running at least 150MHz should be sufficient for a VPN with Internet connections of T1 speed or slower.) Of course, you will need the Windows NT Server 4 operating system itself. The same computer can be used to establish dial-in RAS sessions via modem as well as VPN RAS connections over the Internet, but (as mentioned in previous sections) the computer should not also function as the file and print server for your network or the firewall for your network.

You will also need a connection to the Internet for your LAN. The RAS server does not have to be the computer that establishes the connection; in fact, it is better if a different computer, which runs firewall software, performs that role. In summary, to use RAS to establish a VPN over the Internet you will need:

- A dedicated Pentium–class computer
- Windows NT Server 4 (preferably running Service Pack 3 or higher)
- RAS
- PPTP
- A constant connection to an Internet service provider
- A LAN connection to your network

Client Requirements There are two sets of requirements for connecting remote computers to your LAN via PPTP. If you set up a class of service with the ISP that the client computer will be using and that service includes establishing a PPTP tunnel, the client computer needs only to dial up the ISP using the PPP protocol. The client computer will then be able to do anything that it would be able to do if it had dialed directly into the RAS server on your LAN. Before the client can connect in this manner, however, you will have to negotiate with the ISP to set up the service.

On the other hand, if the ISP does not offer the PPTP service (or if you don't want to use the ISP's service), the client computer must support the PPTP protocol itself. Microsoft provided client software for all versions of Windows and the Mac OS that allows these operating systems to connect to an RAS server via PPTP.

The client computer must also have a connection to an Internet service provider. This connection can be a temporary connection made via a regular modem, ISDN, or xDSL, or it can be a permanent connection made by a cable modem or a leased line. In summary, the requirements for a remote client are as follows:

- Windows (95 or later), or Mac OS
- PPTP–capable dial–in software
- A temporary or permanent Internet connection

Establishing and Securing the VPN To establish a VPN, you need the RAS software and the PPTP protocol on at least one server computer in each LAN. The RAS and PPTP software can be found in the i386 (or Alpha) directory of the Windows NT 4 Server installation CD–ROM. You should add RAS from the Services tab and PPTP from the Protocols tab of the Network Control Panel program.

When you add PPTP to the services supported by NT, you must specify how many Virtual Private Networks RAS will support. You can enter a number from 1 to 256. This number should equal the number of other LANs this RAS server will maintain connections to, in addition to the maximum number of simultaneous remote computer users.

In Remote Access Setup, you will need to add the VPN ports that will appear in the RAScapable devices list. The number of VPN ports that will appear will match the number of PPTP connections that you selected (to be supported).

By default, all of the VPN ports will be configured to only accept connections. To establish a connection to another RAS server you will have to configure a VPN port for a dialout connection. In any pair of communicating RAS servers, one must have a port configured for a dial–in connection and the other a port for dial–out connection. When establishing a dial–out connection to another

RAS server, you will also have to create a phone book entry so that RAS can make the connection.

You should find the security features for VPN ports familiar; they are the same as the security features for any other RAS connection. As with regular dial-in connections, you can configure which protocols may be used for dialing out, which protocols may be used for dialing in, and the encryption settings for VPN communications. You should require Microsoft-encrypted authentication and data encryption.

Since PPTP can support more than one transport protocol (NetBEUI, TCP/IP, and NWLink), the use of PPTP for your VPN doesn't limit you to using TCP/IP for your file and print services. Consider using a different protocol for file and print services in order to make it more difficult for network intruders to penetrate your security.

PPTP Vulnerabilities The PPTP protocol as implemented by Microsoft has some problems you should be aware of before you implement your VPN using Microsoft RAS servers. (Counterpane Systems has a white paper exploring them in detail at <http://www.counterpane.com/>.) Some of the weaknesses are easily dealt with by a diligent network administrator; others are less easily solved. The sections that follow cover some of the topics you should be aware of.

LAN Manager Authentication Makes Password Cracking Easy

Windows NT supports LAN Manager authentication and Windows NT password authentication to allow older network clients to connect to Windows NT networks. Windows 95 provides both LAN Manager and Windows NT-style passwords to support connection to older LAN Manager servers and to Windows NT. The problem is that LAN Manager passwords are simplified (reduced to eight characters or less and shifted to one case) before being presented to the server. A network intruder that eavesdrops on the PPTP authentication process or redirects the remote client computer in order to capture the password can easily crack the LAN Manager password and use that simplified password to gain access to PPTP servers supporting LAN Manager authentication. With a bit more processing, the intruder can use the simplified password to determine the full password.

Tip Disable LAN Manager authentication on both the PPTP client(s) and server(s). (In Windows 98 LAN Manager authentication is disabled by default.)

Microsoft's 40-Bit Encryption Is Weak and Flawed

With Windows 95, 98, and NT you have the option of using 40-bit encryption or 128-bit encryption for authentication and encrypted communication. In addition to being weak (the 40-bit key makes the encrypted communication relatively easy to crack using brute-force cryptanalysis), the 40-bit protocol used by Microsoft does not salt (modify with a random number provided by the server) the key used to establish the session. The key is simply generated from the LAN Manager hash of the user's password, and since the password will not change from one session to the next, neither will the key.

On the other hand, the 128-bit encryption does salt the key with a number provided by the server, thereby resulting in a different key for each session. Note, however, that the key is still based on a hash of the user's password, and most passwords contain much less than 128 bits of randomness (unguessability). Passwords with non-alphanumeric characters in them are much more difficult to crack than short, alphanumeric passwords.

Tip Use only 128-bit encryption on PPTP clients and servers. Require passwords with non-alphanumeric characters for PPTP users.

PPTP Clients Are Vulnerable to Internet Attack

The remote PPTP clients to your network have two network connections—one network connection to their ISP and another (through the ISP) to your network. You must make sure that hackers can't penetrate the client computer through the IP connection at the ISP and then come in through the PPTP connection established by that client (or establish a PPTP connection of their own after having captured the passwords and network configuration information stored on the client).

A properly secured client will not export network services that can be compromised by network intruders, such as web or FTP servers. By no means should the client have file and print sharing enabled; Internet users would be able to see the NetBIOS ports as well as the members of your VPN. The Internet client software (web browsers and mail software) should be kept up to date; bug fixes and security updates should be applied promptly.

If possible, get strong client-based software that puts a stateful inspection firewall on every PPTP client. This type of software is new but increasingly available.

Control Data May Be Intercepted

PPTP has a problem for which there is no easy solution. While the data exchanged by the client and server are encrypted, some control information used to establish the session is not. Information that can be gleaned by an eavesdropper includes:

- Client and server IP addresses
- Number of PPTP virtual tunnels available on the server
- Client RAS version
- Client NetBIOS name
- DNS servers handed to the client by the server
- Client username and the password hash

There is no quick fix to this problem other than to use a different software or hardware package for establishing your VPN. PPTP control channel spoofing can crash the PPTP Server; an additional vulnerability of PPTP is that it is easy to crash an RAS server by sending it spurious and wrong PPTP control information. Your only defense against this sort of attack is to limit who can communicate with your PPTP server by implementing connection restrictions in your firewall.

Tip Use your firewall to restrict the IP addresses or address ranges that are allowed to connect to your PPTP server. Drop source-routed frames to make IP spoofing of valid addresses more difficult.

PPTP Filtering When you install the PPTP protocol there will be a new check box that shows up in the TCP/IP Properties settings. (To access these settings go to the Network Control Panel, click the Protocols tab, select TCP/IP, click the Properties button, and then click the Advanced button.) The Enable PPTP Filtering option gives you the ability to restrict the client traffic that will pass through the RAS server by the IP address of the client. The option by itself doesn't give you a great deal of control over what is filtered, but by adding a key and two values to the RAS server's registry, you can explicitly list the allowed IP addresses for PPTP clients.

PPTP filtering does not protect you from denial-of-service attacks that use the PPTP control channel to crash your RAS server. IP restrictions more properly belong in your firewall.

Encrypted Authentication

Authentication in a Windows NT system consists of shared secret authentication by providing an account name and a password. The password authentication mechanism is performed through a hashed challenge–and–response mechanism so that the password is never transmitted on the network.

Unfortunately, Windows NT passwords are limited to a maximum length of just 14 from a selection of 96 possible characters. Since most people tend to choose words they can remember easily, password protection is not sufficient for Internet–based authentication.

Consider the following scenario: the commonly available NetBIOS Auditing Tool can perform automated password attacks against a Windows NT server at a rate of one password attempt per second (and at about 50 per second on a workstation). Assume that a hacker actually wanted to compromise the administrative account of your machine and used ten simultaneous NetBIOS Auditing Tool sessions from a single machine to do it. Table 15.1 lists how long it would take to crack various types of passwords.

Table 15.1: Time required to crack common sets of passwords

Password Set	Members in Set	Time to Crack
Most common passwords	50	5 seconds
Slang word	200	20 seconds
First name	7,500	13 minutes
Last name	40,000	67 minutes
Common English (CE)	25,000	42 minutes
English	750,000	21 hours
Top 10 common languages	250,000	7 hours
Name + any character	4,560,000	126 hours
CE + any character	2,400,000	66 hours
CE + any character + CE	60,000,000,000	190 years
Completely random 14 char	5.6×10^{27}	17 quintillion years
*Top 10 common languages include the 25,000 most commonly used words of the 10 most common languages. This is why the set is smaller than the complete English language.		

Notice that at ten attempts per second, a completely random 14–character password would take a billion times longer than the universe has existed to crack. This is long enough, and statistics like these led Microsoft to believe that a 14–character limitation on password length would be sufficient.

But humans are not computers. The vast majority of passwords (my most commonly used password, in fact) can be broken in less than a single workday because they fall into very small known sets like those shown in Table 15.1. My experience tells me that humans simply will not accept memorizing random garbage longer than a telephone number or a social security number, and that they're simply not capable of reliably repeating that task once a month when the system invalidates their old passwords.

To compensate for these problems, I recommend linking two randomly chosen words and salting the combination with any other random character. The resultant password is easy to remember and sufficiently difficult to crack to satisfy most security requirements.

Passwords in Windows systems are further compromised by a number of extremely poor convenience choices that Microsoft has built into their client operating systems and applications:

- Windows 95 and 98 store all entered passwords in password files that are easily pilfered from nonsecure client computers and trivial to decrypt.
- Internet Explorer will automatically respond to challenges for an encrypted password from a web server—yielding a decodable hashed value to the server because the server knows the random seed value it provided for the hash. This is a serious flaw in all challenge-and-response systems, but is made especially bad in the case of Internet Explorer because the entire negotiation occurs without the user's awareness.
- Backward compatibility with LAN Manager's far weaker authentication system is built into Windows NT and difficult to disable.

More disturbing than the presence of these features is the fact that Microsoft does not allow the user to eliminate or disable them in favor of security.

Note Windows 2000 allows passwords of up to 256 characters, which considerably improves the security of the operating system if consistently used.

Limitations

The firewalling functions of Windows NT 4 are not sufficient to provide border security for a number of reasons:

- NT's packet filtering is simplistic, providing only minimal stateless functionality.
- There is no NAT or proxy service, so internal hosts are not hidden.
- The tunneling protocol relies upon shared secret passwords that are generally easy to discover and use for other services.
- The authentication service, while reasonably strong theoretically, suffers from numerous well-known exploits and is weakened by its default support for weaker authentication.
- There is no specific security-based logging and alerting mechanism, although the operating system's strong support for logging and alerting can be used to compensate for this deficiency.
- There is no managed method to propagate consistent security policy in an enterprise beyond authentication policy, which is automatically handled through the single logon functionality of domain security.

Most of these limitations have already been discussed, but the important ones are reiterated here.

No NAT

Windows NT does not currently include the ability to perform Network Address Translation, although this capability is available in Windows 2000. This means that if you cannot complete your connection requirements using a proxy server, you must allow routing from the Internet inside your private network using public numbers.

No Proxy

Windows NT does not include proxy services, although Microsoft Proxy Server, an add-on BackOffice component designed for Windows NT, is a reasonably good proxy server application. However, despite Microsoft's use of the word in their advertisements, it is not a firewall because it does nothing to protect the source operating system from denial of service or intrusion.

Limited Logging and Monitoring

Microsoft's security monitoring is spotty and incomplete because it was designed to support more generalized operating system monitoring and logging. Logging and monitoring is only performed on higher-level operating system objects like files and user accounts. Indicators of hacking activity like malformed packets or source-routed packets are impossible to track in Windows NT.

Performance

Routing performance in Windows NT is lower than that of most Unix implementations on the same hardware, and lower than most comparably priced dedicated routers. Windows NT's routing performance is perfectly sufficient for routing to the Internet, however, because any Internet link that a server is directly attached to is bound to be slower than what the server could comfortably handle.

Enabling packet filtering does not put an appreciable load on an NT server. Because the filtering is stateless, it does not consume any more memory than the routing function itself. The tunneling performance is also lackluster, but sufficient for most Internet connections.

Windows 2000

Windows 2000, the current successor to all Microsoft operating systems, is based on the Windows NT kernel with the interface of Windows 98. The operating system supports many convenience features missing in Windows NT but present in Windows 98, such as USB support, IEEE 1394 (Firewire) support, and support for hot swapping PCM-CIA cards.

Note At the time of this writing, Windows XP Professional is available, but no server version has been announced. As far as security features are concerned, there is no significant difference between Windows 2000 Professional and Windows XP Professional.

None of the interface features are especially important for server functions, but the Server packages also include the following network-related services that, in addition to those provided by Windows NT 4, are very relevant to firewall operations:

- CryptoAPI/Public Key Infrastructure
- Kerberos authentication
- Network Address Translation
- Improved Packet Filtering
- IPX Packet Filtering
- Layer-2 Tunneling Protocol (L2TP)
- IPSec

The only missing feature is network proxying, which is provided by the addition of Microsoft Internet Security and Acceleration Server (a BackOffice product). Each of the supported firewall features of Windows 2000 is discussed in the following sections. Proxy servers are discussed in Chapter 8.

Windows 2000 Professional Edition, the direct successor to NT Workstation, includes a lightweight Network Address Translator designed for sharing dial-up and consumer Internet subscriber line technologies like xDSL and cable modems. The feature allows you to install two network adapters and automatically establish NAT and a DHCP server for interior clients.

CryptoAPI

CryptoAPI is a set of operating system routines that make any encryption algorithm look the same to the operating system and other programs. This means that you can install any cryptographic module (called a Cryptographic Security Provider, or CSP), such as DES, RSA, Blowfish, or GOST into Windows to be used pervasively. It also means that as existing algorithms are weakened by greater computing power or found to be flawed, they can be replaced easily and completely. CryptoAPI was actually released in Windows NT Server 4 Service Pack 3, but had no important effect on that already functional operating system. It is an integral part of Windows 2000.

For example, the Encrypted File System of Windows 2000 relies upon CryptoAPI to perform key and certificate generation, encryption, and decryption. So, although by default it uses the RSA encryption for key generation and the RC4 stream cipher for bulk data encryption, you could replace the default RSA CSP with a Blowfish CSP to change Windows NT's EFS encryption to use the Blowfish cipher. This not only gives the end users complete control over the encryption methods, it allows them to create their own encryption modules if they feel the need.

CryptoAPI passes generic service requests from applications to the various installed CSPs to perform the following functions:

- Public key generation
- Encryption/decryption
- Digital signing
- Hashing

There's no reason that the CSP has to use the same algorithm for each of these functions; in fact, the default CSP does not. A CSP need not perform all of these functions because multiple CSPs can be installed in the system and used for different purposes.

Public Key Infrastructure (PKI) is Microsoft's term for the pervasive changes that CryptoAPI allows. Nearly all security services in Windows 2000, including EFS, IPsec, and RAS authentication, rely upon CryptoAPI.

Windows 2000 comes with default CSP written to implement RSA for public key generation, RC2 for bulk block encryption, RC4 for bulk stream encryption, and MD4 and MD5 for digital signing and hashing. This CSP can be set up to use the local registry, a group security policy, or a key server for the storage of public keys.

Warning Storing your cryptographic keys in the registry of the protected machine makes it possible for someone who has seized your equipment to extract the key and decrypt your encrypted content. Always store cryptographic keys on a foreign server or removable device.

Kerberos Authentication

Kerberos authentication—an Internet standard for user authentication—is the basis for the new security features that became available with Windows 2000. Like the Windows NT domain model,

Kerberos is a trusted authentication system, meaning all the servers in the domain hierarchy use (and trust) the same system.

Kerberos does not rely upon a secure network, the physical security of network clients, or the host's IP address for security. Kerberos was designed from the ground up with the assumption that traffic on the network could be read, written, and changed at will by a hacker who was theoretically perfect and who would understand all of the security-related issues in the network and the Kerberos system. Kerberos does not rely on security through obscurity at all.

Kerberos keeps a database of the private keys of its clients. Those clients may be users or network services. If the client is a user, that private key is an encrypted password. In this respect, the system behaves very much like NT security.

Once a Kerberos client has been authenticated by the Kerberos system, the Kerberos server generates unique session keys that two clients (the user and the service being used) use to authenticate messages between one another. Kerberos supports three levels of encryption security:

- **Authentication** Proves to each client that the other is who it says it is initially, but provides no further identification midstream.
- **Message Signing** Includes an encrypted signature with each packet of data, proving to each client that the message originates from the other client and is not a forgery. Message signing is often referred to as safe messages in Kerberos documentation.
- **Encryption** Makes the contents of each packet indecipherable to parties other than the clients engaged in the conversation. These are called private messages. They are used by the Kerberos system for the transmission of passwords over the network. In most Kerberos systems, any or all of these methods can be used for any session, depending upon the level of security required and the amount of load that the system can tolerate for the encryption process.

Kerberos is based on DES encryption. DES (Data Encryption Standard) was developed by the U.S. government for public use. That being the case, many people have theorized that the government may have some method for decrypting the contents of DES-based systems that is faster than a simple brute-force attack. This has not been proven, and no statistical evidence has implied any abnormal weakness in DES security. Kerberos is not dependent upon DES encryption

Kerberos session keys have a short valid lifetime. If an intruder gains access to a session key, it's only useful until it expires. If your session key's lifetime is set to a reasonably short value, such as one day, even if compromised, it can only be used for that day. Copying a valid session key and attempting to reuse it is called a replay attack because the session key, although not decrypted, is simply used again by a third party to gain access. Most Kerberos implementations attach to each message a time stamp that must be valid to within a few minutes or the message is assumed to be an attempt at a replay attack.

Kerberos also allows authentication to take place using public keys rather than an account name and password. This means you could generate keys that associated businesses could install in their systems to prove their identities to your network and allow whatever limited access you've defined for that key. Kerberos' controllable public security is an important part of Microsoft's long-term strategy for Windows.

Network Address Translation (NAT)

Support for Network Address Translation in Windows 2000 Server versions is strong and very easy to configure; in fact, it's the easiest NAT configuration around. Because the NAT is built into the router software, there's no silliness to deal with concerning routing table additions, specific interface problems, or any of the other issues you deal with when using NAT software that sits above or below the router layer. With Windows 2000 Server, you simply define the range of internal addresses you want to translate for, and you're done.

The NAT software allows reverse translation mappings so you can configure each port on the NAT to map to a specific internal host and port. This makes it easy to create a DMZ using Windows 2000 to shunt Internet services to a group of hosts behind your firewall.

The NAT software also allows for static internal mappings so you can assign a public IP address to an internal host. This feature allows you to make services on that host visible to the external network. It is generally more secure to translate services on a service-by-service basis using the IP address of the firewall, however.

NAT in Windows Workstations

Windows 2000 Professional, Me, and XP support a limited version of NAT called Internet Connection Sharing, which allows you to specify that a network adapter on the public should be "shared" to the adapter in the private network. The private network adapter's address is then set to 192.168.0.1, and DHCP is automatically provided for hosts in the private network to put them in the 192.168.0.0 network. You cannot configure the NAT network range, but you can enable port forwarding to machines inside the private network.

Network Load Balancing

Windows 2000 Advanced Server includes a network load-balancing feature that can be enabled on a per-interface basis. Network load balancing allows a group of servers to provide services using the same public IP address and share the load on a fair share basis. You can assign various load weights to servers in the group based on their various service capacities, or you can specify that all hosts should be loaded equally.

Because Windows 2000's load-balancing feature is performed by all hosts simultaneously, there's no requirement that all data stream through a single NAT host. This makes it possible to service extremely busy network services that would overwhelm the routing capabilities of a NAT. All hosts in the IP cluster must be able to receive routed data to the shared IP address and must be able to communicate with one another for the load balancing to operate correctly.

Windows 2000 Advanced servers with more than one network interface adapter can use one adapter for the shared IP address and use the other to negotiate for connections with the other load balanced servers. This allows the balancing negotiation traffic to be kept separate from the actual traffic being balanced, reducing contention and increasing overall throughput.

Windows Load Balancing can interact badly with switches or Layer-2 routers because of the way NLB works. NLB creates a virtual MAC address that is shared by all members of the load-balancing cluster. Switches, on the other hand assume that every MAC address is associated with just a single port on the switch. The load-balancing service also expects Ethernet broadcasts to be seen

by all of the participating load-balancing servers. Because the switch may "learn" that the group MAC address belongs to a specific port and send all traffic to it, NLB may fail to operate properly. Although Microsoft lists a few hacks to change the way that NLB works and recommends modifying the configuration of your switch to get around these problems, by far the easiest solution is to simply use a hub in front of a load-balancing group or put your switch in hub mode by creating a VLAN group including all the nodes in the cluster.

Improved Packet Filtering

Packet filtering in Windows 2000 is much improved, but it remains stateless. Windows 2000 is capable of performing both incoming and outgoing packet filtering, as well as specific protocol blocking and specific allowance. Configuring the IP filter is easier in Windows 2000 than in Windows NT 4, once you're familiar with the Microsoft Management Console (MMC) that is used to administer all administrative features of the operating system. Because the packet filter is stateless, it is still not as strong as the filter in a true stateless inspection firewall.

The packet filter also allows filtering of various ICMP message types, so you can protect against certain denial-of-service attacks while retaining the useful services of the ICMP protocol.

IPX Packet Filtering

Windows 2000 supports filtering of IPX packets, even though Microsoft is trying to deprecate support for IPX. This deprecation is a shame, because IPX is an excellent transport inside a network while WinSock compatible proxies are available to perform TCP/IP translation at Internet borders.

IPX packet filtering allows basically the same level of packet-filtering control as the IP filter.

Layer-2 Tunneling Protocol (L2TP)

Layer-2 (Data Link layer) Tunneling Protocol is a derivative of PPTP refined by Cisco Systems to create a tunneling protocol that is independent of IP. PPTP can only be carried inside IP packets; any connected frame forwarding mechanism can route L2TP. One benefit of the decoupling between the tunnel and the transport is that L2TP can support multiple tunnels between the same two end points, whereas PPTP supports only one. One use for this functionality might be to create different tunnels for different quality of service requirements, or to exploit two physical circuits to create one "bonded" circuit with different end points.

Microsoft's implementation of L2TP uses IPSec encryption to encrypt the payload. Microsoft's implementation of L2TP can be accurately thought of as PPP with an IPSec payload.

IPSec

IPSec is a Layer-3 (Network layer) encryption technology. It provides encrypted transportation services similar to PPTP and L2TP, except that there is no concept of a connected session or tunnel in the Windows 2000 implementation. Windows 2000's IPSec support is limited to Transport mode encryption—it does not support tunnel mode and so is not appropriate for creating virtual private networks. L2TP or PPTP can be used for that purpose, however.

In IPSec transport mode, individual encrypted packets can be transmitted between hosts, and it is simply assumed that some prior authentication has occurred that will enable the hosts to decrypt the packets when they arrive. This is entirely different from the concept of tunnels, which are sessions

between two machines that maintain the state of the connection. When the tunnel is closed, the information necessary to decrypt the tunnel's contents is gone. IPSec is merely IP with encrypted payloads; there is no encapsulation within another protocol.

Because there is no overhead for tunnel establishment or maintenance, IPSec is more appropriate for message-oriented communications than either L2TP or PPTP.

Case Study: Windows as a Firewall?

Can Windows really be hardened enough to act as a firewall without third-party security software? No. At least, not the way Microsoft is currently approaching the problem.

Windows NT 4 suffers from a number of well known TCP/IP stack weaknesses, most of which have been shored up as of its last "hotfix rollup" (apparently the term of choice for a service pack after Microsoft has declared that there will be no further service packs). Windows 2000 fares quite a bit better, but still relies upon a constant stream of updates to remain secure as new threats are found.

There are four major problems with relying on security updates to keep a firewall secure:

1. Updates are not automatic, which means that lax security administrators or companies without proactive security administration will fall behind (if updates were automatic, you can bet that some hacker would figure out how to exploit the automation mechanism to push a Trojan horse to all subscribers).
2. Updates always follow exploits by between three days and three weeks, depending upon the complexity of the update. This means that there's always a window of vulnerability between the discovery of an exploit and the release of its patch, during which you are vulnerable.
3. Microsoft will eventually release a patch that causes crashing problems. They've been lucky to date, but there's no way that they can sustain the rapid-fire security releases to code that executes within the kernel and takes into account every possible configuration every time. Sooner or later, they're going to release a hotfix that causes widespread crashing problems for their users.
4. Machines must be connected to the Internet to receive the latest updates, and can be exploited while they are downloading patches. Sound theoretical? During the final production of this book, we were setting up a public server for a client that was scanned and hit with the NIMDA worm seven minutes after we brought it online, while we were running Windows update from Microsoft's website to patch against that and other defects. To defend against this problem, place new machines behind firewalls until after they're running the latest patches.

What's wrong with Windows that it requires all these hotfixes? Why can't Microsoft "get it right?" The answer to this question comes down to motivation: Microsoft engineers Windows to fulfill a set of user feature requirements, and then releases the software because they're under a shipping deadline. The software is large and complex, and there is no real room in the schedule to test for esoteric problems. This leads to the constant countermeasures hotfix cycle that Microsoft is locked into now.

The difference in motivation explains why "hardened" operating system vendors like firewall manufacturers and OpenBSD can secure their systems against attacks in advance, because the only feature they're working for is security. Their operating systems are orders of magnitude simpler than Windows 2000, and they aren't trying to stuff every feature under the sun into their kernels. The only feature they need is security, so they can proactively comb through the smaller code-base of their operating systems every time vulnerability hits any operating system to ensure that it, or any

variant of it, won't affect their operating system.

Microsoft web and mail servers are not safe on the Internet without the protection of a firewall in front of them and without security proxies to filter the protocols they provide. This Achilles heel will continue until Microsoft finally caves in to the bad press and either buys or develops true enterprise policy-based firewall software and includes it for free and with a strong default security policy in the server versions of Windows. I see this happening by 2003, as security problems hamper the deployment of .NET.

Linux is safer than Windows because the standard distributions have already done this. IPChains or IPTables (depending upon the distribution) is automatically configured during the installation process of the latest versions of the major distributions, according to the settings you configure during installation. This is one of the major reasons that Linux is more secure than Windows.

Chapter 16: Open Source Firewalls

Overview

This chapter could also be titled "How Hackers Protect Themselves." Hackers have computers too and want to keep their own machines free of intrusion from the Internet. Paradoxically, these computers may be the most secure computers on the Internet, because the hackers use free software that they can examine for security problems, they are the first to discover (or create) security weaknesses, and they fix their own systems as soon as loopholes are discovered. They share the security fixes with anyone who wants them, and you can take advantage of this fast response to new threats in your own network by using the same software they do.

There are several free software packages that you can use to craft a firewall for your network. Most of these packages run on free operating systems such as Linux, BSD, or DOS. We'll examine several popular packet-filtering packages—IPChains or IPTables for Linux, Drawbridge for FreeBSD, and ipf for OpenBSD. For proxying we'll check out the Squid web proxy and the Trusted Information System's Firewall Toolkit (TIS FWTK). Finally, for those averse to multitasking, we'll look at IPRoute on top of DOS.

Free firewalls often suffer from a few of the same class problems:

- Weak or missing logging and alerting features
- No real-time firewall monitoring capability
- Weak or missing graphical user interface
- Difficult command prompt-based configuration

These problems all stem from the fact that the software is developed primarily by a single individual or small team rather than by a corporation. Small teams don't have the time or money to spend on ancillary problems like ease of use or sophisticated alerting and logging mechanisms. These features, when present, are nearly always provided by an add-on package developed by a different developer. Free software is developed for people who deeply understand the problem to be solved and the operating system upon which the software runs.

With network security being a strong point but ease-of-use being a weakness of these free packages, several companies have built businesses around "finishing" the free firewalls and selling the result. If you look closely, you'll find that under the hood of many of the commercial firewalls reviewed later in this book lurks a Linux or BSD kernel running IPChains or ipf. For plug-and-play security that may be the better solution, but if you're not afraid to roll your own firewall, then this chapter's for you.

Linux and IPChains or IPTables

Linux is an Internet phenomenon that baffles many people in the traditional computer software industry. Linux was an educational project of a Finnish college student named Linus Torvalds that has gone much further than originally expected. Using free software tools developed for Richard Stallman's GNU (Gnu's Not Unix) project, he wrote a simple operating system for his computer because he didn't like DOS and Windows, and nothing else was available for which he could get source code. He posted his code to the Internet and other people got interested, suggested changes, and (most importantly) sent their own additional code back to Linus to extend the little operating system. Now the little operating system is as fully complex and powerful as Windows NT

or traditional Unix, and all of the source code is still available for anyone to browse, change, and fix for security problems.

Note The biggest security advantage to Linux, according to Linux advocates, is that once a security problem in the operating system is understood, it can be diagnosed and fixed in hours or minutes, and a security patch will be posted to the Internet minutes later. Software or detailed instructions for stopping any intrusion threat is typically made available through security newsgroups and websites within a day of the threat being found. Contrast this to the time it usually takes commercial operating system companies to acknowledge a security problem, prepare a work-around, and then deliver an operating system update (hint: it usually takes longer than a day).

A significant addition to the Linux core operating system is the ability to perform packet filtering and Network Address Translation in the operating system itself. Originally called IP Masquerade because of its NAT ability, the system is now called IPChains or IPTables (depending on the version you're using—both are present in current distributions and either or both can be used), because it allows the administrator to set up chains or tables of rules that a packet must satisfy when it arrives at the Linux computer, is routed within the computer to another adapter, or exits the computer to another network.

IPChains and IPTables provide NAT and packet filtering. Protocol inspection must be provided by a higher-level service. TIS FWTK (described later in this chapter) is an excellent proxy server package that interoperates well with IPChains on Linux, as well as with ipf for OpenBSD. Another less complete but easier to install for Linux option is to use Squid, which is a good HTTP proxy. Other stream-based protocols (such as SMTP, NNTP, POP, and DNS) can just be redirected to other computers using IPChains, which redirects socket connections according to rule sets you define but does not examine the redirected stream to ensure that it is in fact an SMTP, NNTP, POP, or DNS transaction.

Major Feature Set

Linux with IPChains or IPTables supports the following major features:

- Packet-filtering rules are applied to every packet as it arrives, transits the Linux routing stack, and exits. IPChains is stateless; IPTables is stateful. This is the primary functional difference between them.
- Proxies may be established by protocol-specific content filters provided by higher-level services such as TIS FWTK, Apache, or Squid.
- Network Address Translation (dynamic or static) is performed for packets transiting the routing stack to hidden, internal networks.
- DMZs can be established either by filtering access to an externally visible protected subnetwork or by redirecting virtual public addresses to address-translated protected hosts.
- VPN firewall-to-firewall and firewall-to-remote client options are provided as additional Linux components that can be downloaded from the Internet at no cost.
- Port redirection is natively provided by IPChains and IPTables.
- Clever use of IPChains or IPTables with Squid or FWTK can give you transparent proxies.
- Linux with FWTK easily gives you reverse proxies (HTTP, SMTP, FTP, etc.) as well.
- Linux with installable packages such as PoPToP or FreeS/WAN gives you platform generic VPN options (PPTP, IPsec, etc.)
- Additional packages convert the regular Linux syslog reporting system to store logging information to databases and give you e-mail notification.

Minor Feature Set

Linux with IPChains or IPTables supports the following minor features:

- Performance of Linux packet firewalling is fast because standard desktop processors are far faster than the processors used in most dedicated firewalls, and because Linux doesn't have nearly as much networking overhead as most general purpose operating systems. Since it is integrated with the Linux IP stack, the packet filter avoids the overhead of other firewalls that are implemented as user-level programs. This solution can easily handle a busy LAN connection to the Internet even with Network Address Translation enabled.
- Command-line based configuration requires more skill in administration, but allows you to store policies in text files and use scripting tools for dynamic policy management. Many Linux distributions provide a graphical interface that drives the command-line programs, making the software easier to set up and configure.
- Remote management (using a secure shell (SSH) or a web interface) allows you to manage your firewall from other computers in your LAN.
- The packet-filtering rules allow you to use NAT and forwarding for sockets to redirect traffic for particular services (such as HTTP, SMTP, and POP) to protected internal servers.

Security

Linux filters packets before they are delivered to the IP stack for processing, allowing you to protect your computer from malformed packets and other IP-level attacks. Linux provides the full range of options for packets filtering on: the SYN bit, source and destination IP addresses, source and destination ports, packet type, and most other TCP/IP header data elements. Network Address Translation is built into the packet filter, so you can use the same rules to specify the kinds of packets that will be translated and the kinds that will not.

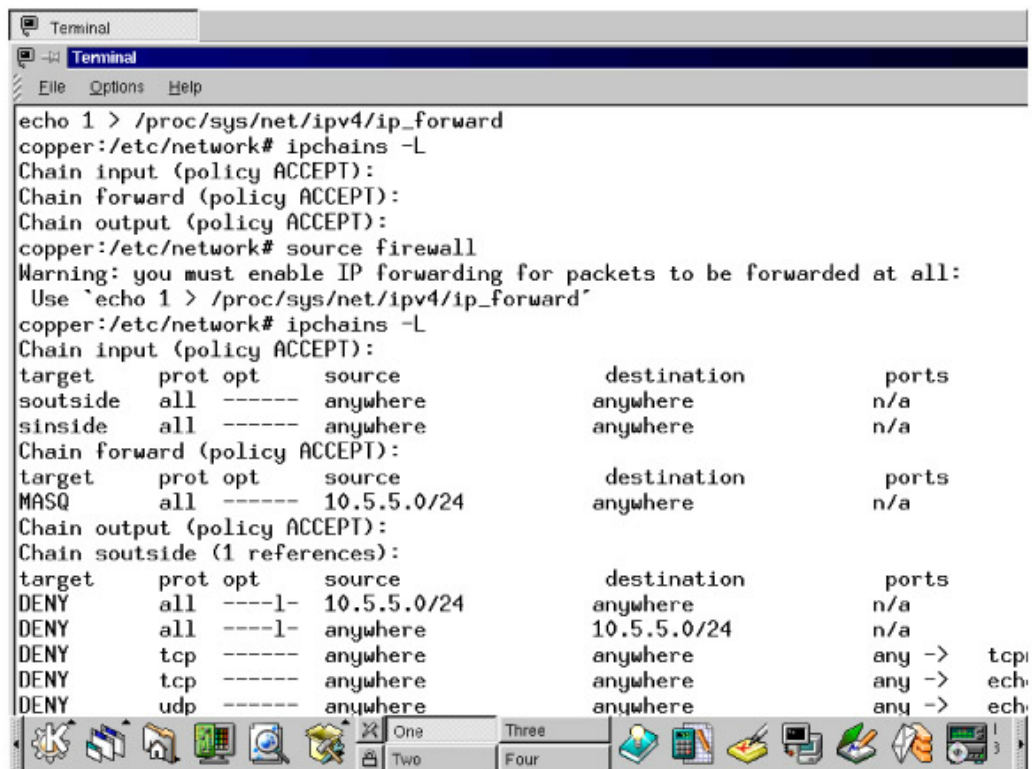
Since Linux does not inspect the data portions of the packets it manipulates, you will need a proxy server to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example). See the sections on Squid and the TIS FWTK package later in this chapter. Also, many web servers will also act as HTTP proxies, and you can use servers for store-and-forward protocols (such as SMTP and NNTP) without modification as protocol proxies for their services. Ideally, you will run these services on a different computer than your firewall, and you will use address translation to redirect the appropriate traffic to and from these servers.

The Linux filtering software evaluates every packet received by the network adapters in the firewall computer according to a set of rules you established when you installed the software. The rules are applied in order, one at a time, until the filter finds a rule that matches the packet and specifies a terminal action, such as ACCEPT or DROP. Since the rules are applied in order, it is vitally important to craft the rules in the right order.

A useful feature of IPChains (and the feature that gives it its name) is the bundling of sets of rules into chains. IPChains starts out with three—INPUT, FORWARD, and OUTPUT. You can establish additional chains and use a rule in INPUT, FORWARD, or OUTPUT to direct packet analysis to the appropriate chain for that type of traffic. This structured rule management makes it easier to reason about the security of the firewall and thereby makes it easier to secure the firewall. IPTables works similarly, but provides stateful inspection.

Interface

Linux packet filtering is administered using the `ipchains` or `iptables` command, which takes as its arguments the rules to be established or modified in the packet filter. Figure 16.1 shows an example of the IPChains command–line syntax.



```
Terminal
File Options Help
echo 1 > /proc/sys/net/ipv4/ip_forward
copper:/etc/network# ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
copper:/etc/network# source firewall
Warning: you must enable IP forwarding for packets to be forwarded at all:
Use `echo 1 > /proc/sys/net/ipv4/ip_forward`
copper:/etc/network# ipchains -L
Chain input (policy ACCEPT):
target    prot opt    source      destination    ports
soutside  all  ----- anywhere      anywhere      n/a
sinside   all  ----- anywhere      anywhere      n/a
Chain forward (policy ACCEPT):
target    prot opt    source      destination    ports
MASQ      all  ----- 10.5.5.0/24   anywhere      n/a
Chain output (policy ACCEPT):
Chain soutside (1 references):
target    prot opt    source      destination    ports
DENY      all  ----1- 10.5.5.0/24   anywhere      n/a
DENY      all  ----1- anywhere      10.5.5.0/24   n/a
DENY      tcp  ----- anywhere      anywhere      any -> tcp
DENY      tcp  ----- anywhere      anywhere      any -> echo
DENY      udp  ----- anywhere      anywhere      any -> echo
```

Figure 16.1: IP Chains command–line interface

Most network administrators who use IPChains create shell scripts to configure the packet filter. The Example IPChains Shell Script sidebar shows you an example of an IPChains packet filter configuration.

Example: IPChains Shell Script

The following example creates a simple, permissive filter policy:

```
#!/bin/sh

ipchains -F

ipchains -A input -p icmp --icmp-type destination-unreachable -j ACCEPT

ipchains -A input -p icmp --icmp-type source-quench -j ACCEPT

ipchains -A input -p icmp --icmp-type time-exceeded -j ACCEPT

ipchains -A input -p icmp --icmp-type parameter-problem -j ACCEPT

ipchains -P forward REJECT

ipchains -A forward -j MASQ -s 10.5.5.0/24 -d ! 10.5.5.0/24
```



```

ipchains -A input -i lo -j ACCEPT

ipchains -A input -i eth0 -p TCP -d 24.0.148.207 81:519 -j DENY

ipchains -A input -i eth0 -p UDP -d 24.0.148.207 81:519 -j DENY

ipchains -A input -i eth0 -p TCP -d 24.0.148.207 530:1024 -j
DENY

ipchains -A input -i eth0 -p UDP -d 24.0.148.207 530:1024 -j
DENY

echo 1 > /proc/sys/net/ipv4/ip_forward

```

Most network administrators would prefer a more graphical and intuitive way to administer their firewall rules, however. Commercial Linux distributors, such as Storm Linux, have bundled graphical management utilities for firewalling and other functions in order to provide extra value to their distributions that you couldn't get from free software. Figure 16.2 is a view of the Storm Linux graphical firewalling tool. The free software isn't far behind, however, and in many cases may be ahead in usability now that they've completed the behind-the-scenes firewall piping to their satisfaction. They're busy polishing the knobs and buttons to make it easy to use and still free to download, as the Gnome firewall configuration tool in Figure 16.3 illustrates.

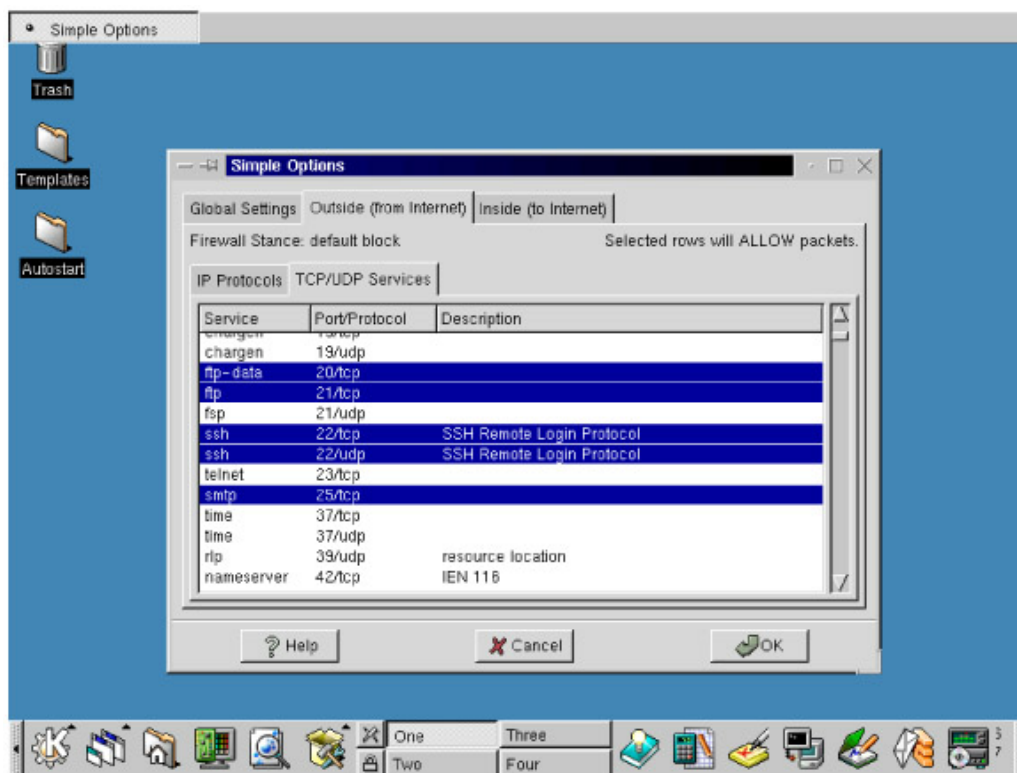


Figure 16.2: The Storm Linux graphical interface to IPChains

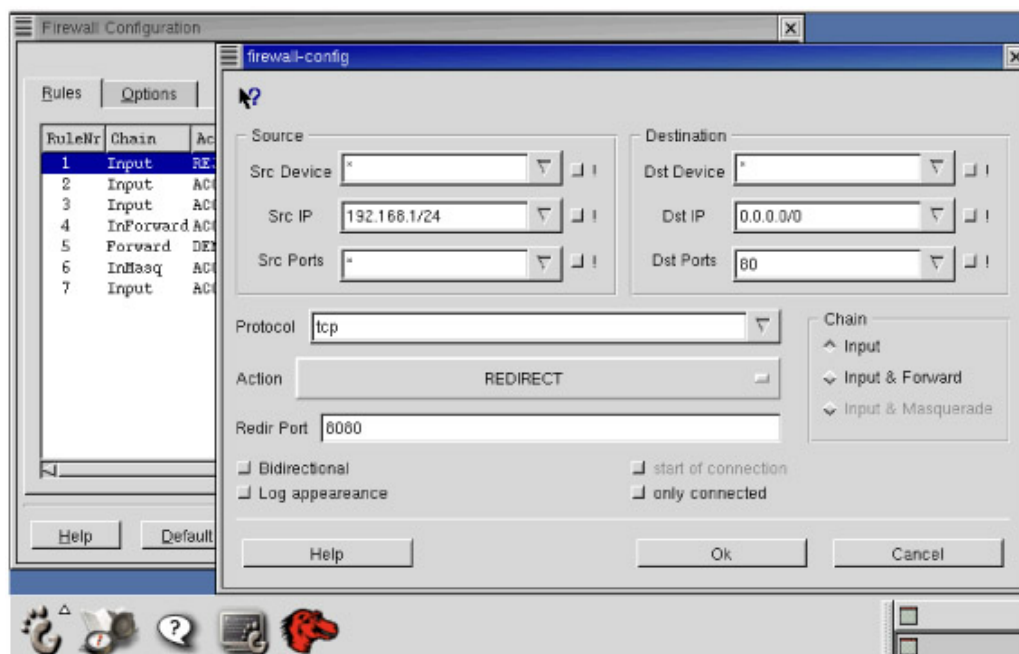


Figure 16.3: The Gnome graphical firewall administration (included in Red Hat Linux 7.2)

Documentation

IPChains and IPTables are documented in the typical Unix manner—readme and install files come with the software installation package (which may already be installed on your Linux system) and documentation of those command-line tools can be found from the man (manual) command.

Tip You can find step-by-step instructions for installing IPChains on the Internet; just search on IPCHAINS from any search engine. Information on IPTables can be found at its site: <http://netfilter.samba.org/>.

Cost and Support

You can find many versions of Linux distributed at no cost. You can also purchase it from several different vendors, including the very popular Red Hat, but you don't have to buy even those copies more than once. Install it on as many servers as you feel like, and protect as many clients as you want. The biggest costs for using Linux are the cost of the computer used to run it and the cost of the time spent by you or the administrator to configure and manage it.

Minimum platform requirements for Linux are easy to meet and should not be expensive:

- A distribution of Linux with IPChains or IPTables
- 486 processor or better, including any processor architecture for which a version of Linux exists
- At least two network interfaces
- 200MB of disk space
- 32MB RAM

The Trusted Information Systems Firewall Toolkit (TIS FWTK)

The TIS FWTK is the granddaddy of freely available firewalls. You can download versions for Linux, NetBSD, Solaris, as well as just about any other flavor of Unix you can think of. If you need a

firewall and you have a spare Unix workstation collecting dust in a corner, this may be the firewall for you.

FWTK was created for the Defense Advanced Research Projects Agency (DARPA) by TIS when DARPA realized that no packet filter would be secure enough to filter protocol content. After fulfilling the terms of their contract with DARPA (which includes making the code public domain), TIS further extended the firewalling concept into a commercial suite known as the Gauntlet Firewall. TIS was acquired by NAI recently, making Gauntlet Firewall a component of the PGP Active Security Suite. FWTK is now maintained by an Internet consortium at <http://www.fwtk.org/>.

FWTK is not a packet filter. Instead it comes with protocol-scrubbing proxies for Telnet, Rlogin, SMTP, FTP, HTTP, and X-Windows. In addition, it comes with a generic TCP passthrough redirector (a SOCKS proxy). FWTK also extends its security controls into the Unix LAN environment, providing centralized network login and resource control using the netacl and authserv utilities.

Major Feature Set

FWTK supports the following major features:

- Protocol-specific content filters are provided by included proxy components.
- Centralized network login and resource access control for Unix computers is maintained using the TIS security mechanisms.

Minor Feature Set

FWTK supports the following minor features:

- Performance of FWTK is relatively fast.
- Command-line based configuration requires more skill in administration, but allows you to store policies in text files and use scripting tools for dynamic policy management.
- Remote management using secure shell (SSH) or a web interface allows you to manage your firewall from other computers in your LAN.
- Address Translation and forwarding for sockets are performed using the FWTK generic TCP plug-board.

Security

FWTK does not filter packets before they are delivered to the IP stack for processing. You must use some other package to protect your computer from malformed packets and other IP-level attacks (IPChains for Linux or ipf for OpenBSD are good choices).

FWTK is a proxy server; it examines the data portions of IP packets to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80 for example). This ensures, for example, that a hacker doesn't use port 80 to access a Trojan horse with its own protocol because your packet filter allows packets in the network to port 80 for HTTP services.

FWTK evaluates data received by the network adapters in the firewall computer according to a set of rules established in its net-perm rule table. The rules are defined according to the port to which the data was sent, while permissions are defined according to the source and destination of the data.

Interface

You enable FWTK by replacing the services to be proxied in the `inetd.conf` (or equivalent service configuration file for your operating system) with the corresponding FWTK filter for that protocol (see "Example FWTK Configuration Entries" in the following sidebar).

Example: FWTK Configuration Entries

This example creates the files indicated containing the text listed after the file.

```
/etc/inetd.conf

### standard config:

#ftp      stream tcp nowait root /usr/sbin/tcpd    /usr/sbin/
wu.ftpd

#telnet   stream tcp nowait root /usr/sbin/tcpd    /usr/sbin/
in.telnetd

### firewall config:

ftp       stream tcp nowait root /usr/local/etc/netacl wu.ftpd

telnet    stream tcp nowait root /usr/local/etc/tn-gw  tn-gw

www       stream tcp  nowait  root   /usr/local/etc/http-gw
httpi-gw

netperm-table

# netacl rules:

# -----

# local FTP users connect to proxy ...

netacl-wu.ftpd: permit-hosts 192.168.* -exec /usr/local/etc/
ftp-gw

# ... while foreigners may use anonymous FTP

netacl-wu.ftpd: permit-hosts *          -exec /usr/sbin/wu.ftpd

# ftp gateway rules:

# -----

# only local users may use FTP proxy

ftp-gw:          timeout 3600

ftp-gw:          denial-msg  /usr/local/etc/ftp-deny.txt

ftp-gw:          welcome-msg /usr/local/etc/ftp-welcome.txt

ftp-gw:          help-msg    /usr/local/etc/ftp-help.txt

ftp-gw:          permit-hosts 192.168.* -log { retr stor }
```

```

# telnet gateway rules:

# -----

# only local users may use Telnet proxy

tn-gw:          timeout 3600

tn-gw:          denial-msg  /usr/local/etc/tn-deney.txt

tn-gw:          welcome-msg /usr/local/etc/tn-welcome.txt

tn-gw:          help-msg    /usr/local/etc/tn-help.txt

tn-gw:          permit-hosts 192.168.*


# http gateway rules:

# -----

http-gw: deny-hosts unknown

http-gw: permit-hosts 192.168.*

http-gw: permit-hosts * -httpd www.mybiz.com


# smap and smapd rules:

#-----

smap, smapd:    userid 20

smap, smapd:    directory  /usr/local/var/spool/smap

smapd:          executable  /usr/local/etc/smapd

smap:           maxbytes 2097152

smap:           maxrecip 4000

smap:           timeout 3600

```

The FWTK proxies read their configuration from the net-perm table, which describes for each protocol those hosts (source and destination) that are allowed to use the proxy. The Example FWTK Configuration Entries sidebar shows sample configurations for FTP, Telnet, and HTTP.

IPChains is administered using the ipchains command, which takes as its arguments the rules to be established or modified in the IPChains packet filter. Its syntax is described in the previous section.

Documentation

FWTK is documented in the typical Unix manner—readme and install files come with the software installation package (which may already be installed on your Linux system) and documentation of the FWTK commands can be found from the man (manual) command.

Tip You can find step-by-step instructions for installing FWTK on the Internet— go to <http://www.fwtk.org/>.

Cost and Support

The FWTK is freely downloadable, and you can use it without licensing fees. The greatest costs are the time it takes to set it up and the time spent by you or the administrator to configure and manage it.

Minimum platform requirements for FWTK (on Linux) are easy to meet and should not be expensive:

- 486 processor or better
- At least two network interfaces
- 200MB of disk space
- 32MB RAM

FreeBSD and Drawbridge

Linux isn't the only free operating system out there—another free Unix operating system shares its tradition of software freely distributed with source code. It is called FreeBSD, and is also used widely (although less visibly) on the Internet to provide network services including web serving, routing, and firewalling. When the Texas A&M University found themselves under concerted network attack from a horde of hackers, they needed a flexible and stable platform that they could build on to create a firewall that fit their specific needs; they chose FreeBSD and they called the resulting packet filter Drawbridge.

Drawbridge, unlike IPChains, is not a part of the operating system; it's a user-level program that directly controls the network adapters of the host computer. Like other packet filters, Drawbridge uses lists of rules to process packets. Since it was developed at a university where all of the computers are expected to be a part of the Internet as a whole, Drawbridge does not perform Network Address Translation.

Where the TAMU software shines, however, is in its ability to report and act on suspicious activity using the components called tcplogger, udplogger, netwatch, and netstat. Universities have lots of interesting computers that are juicy targets for hackers. Unfortunately, the threat doesn't just come from the outside—university network administrators have to protect themselves from attacks from within their computer labs as well. Since universities are based on the precept of fostering communication, not restricting it, a draconian security policy isn't practical. Instead the administrators keep a watchful eye on network activity and stay ready to swiftly respond to threats.

Major Feature Set

FreeBSD with Drawbridge supports the following major features:

- Packet filtering rules are applied to every packet by Drawbridge before being passed to the FreeBSD network stack.
- Protocol specific content filters are provided by higher-level services such as Apache and Jigsaw.
- VPN firewall-to-firewall and firewall-to-remote client options are provided as additional

FreeBSD components that can be downloaded from the Internet at no cost.

- Threat detection and response is performed using the tcplogger, udplogger, netwatch, and netstat tools.

Minor Feature Set

FreeBSD with Drawbridge supports the following minor features:

- Performance of FreeBSD with Drawbridge is fast. It can easily handle a busy LAN connection to the Internet.
- Command-line based configuration requires more skill in administration but allows you to store policies in text files and use scripting tools for dynamic policy management.
- Remote management using secure shell (ssh) or remote control software such as VNC allows you to manage your firewall from other computers in your LAN.

Security

Drawbridge filters packets before they are delivered to the IP stack for processing, allowing you to protect your computer from malformed packets and other IP-level attacks. Drawbridge provides the full range of options for packets filtering on: the SYN bit, source and destination IP addresses, source and destination ports, packet type, and so on.

Since Drawbridge does not inspect the data portions of the packets it manipulates, you will need a proxy server to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example). Many web servers will also act as HTTP proxies, and you can use servers for store and forward protocols (such as SMTP and NNTP) unmodified as protocol proxies for their services. Ideally, you will run these services on a different computer than your firewall, and you will use address translation to redirect the appropriate traffic to and from these servers.

Drawbridge evaluates every packet received by the network adapters in the firewall computer according to a set of rules stored in its database in RAM (loaded when Drawbridge starts). The rules are applied in order, one at a time, until Drawbridge finds a rule that matches the packet and specifies a terminal action, such as ACCEPT or DROP. Since the rules are applied in order, it is vitally important to craft the rules in the right order.

The tcplogger, udplogger, netwatch, and netstat utilities (which don't necessarily have to run on the same computer as Drawbridge) scan all of the packets crossing or passing by the computer they are run on. Sophisticated analysis algorithms identify suspicious activity such as root logon attempts from off-campus, excessive FTP traffic coming from a computer that shouldn't be hosting an FTP server, and so on.

Interface

Drawbridge packet-filtering rules are created in text files and then compiled with the Drawbridge Filter Compiler into the tables used by the Drawbridge packet filter. Instead of entering them one at a time or making a shell script (as you do with IPChains), you type all the rules out at once in a text file, then you compile the file.

Because universities are heterogeneous computing environments with many special cases about what kind of traffic can go where, packet filtering rules for them are often large and complex. Drawbridge rules files reduce the complexity by allowing you to specify groups of permissions and

groups of affected computers or subnets. This way you can have one set of rules for department web servers, for example, and another for the various open-access computer labs on campus.

Example: Drawbridge Filter Source File

```
#----- Group Definitions -----#

#

# Make some useful definitions

#

# Allow all TCP out and no TCP in except:

# src=ftp-data, smtp, auth, gopher, and www

define tcpdefault      <1-65535/tcp out>, <src=ftp-data/tcp
in>,

                        <smtp/tcp in>, <auth/tcp in>,

                        <gopher/tcp in>, <www/tcp in>;

# Allow all UDP except ports 7-19, tftp, sunrpc, snmp, xdmcp,
and nfs

# Note: we don't use '!' to exclude ports because it can't be
overridden

define udpdefault      <1-6/udp in>, <20-68/udp in>, <70-110/
udp in>, <112-160/udp in>, <162-176/udp in>, <178-2048/udp
in>,

                        <2050-65535/udp in>;

# Allow all known ICMP except redirect

define icmpdefault     <0-4/icmp in>, <6-18/icmp in>;

define telnet          <telnet/tcp in>, <ftp/tcp in>;

define popmail         <109-110/tcp in>;

define blockall        <!1-65535/tcp in-out>, <!1-65535/udp
in>,

                        <!0-255/icmp in>;

#

# The special name 'default' defines access for hosts not
listed in this file

#

define default          icmpdefault, udpdefault, tcpdefault;
```



```

#----- Table Definitions -----#

#

# Reject all incoming packets with a source address of local-
host or broadcast.
#

reject 127.0.0.0          255.0.0.0;

reject 0.0.0.0           255.255.255.255;

reject 255.255.255.255  255.255.255.255;

#

# Reject all incoming packets with a source address that is
within our # class C network.  Make an exception for our out-
side router.

#

reject 192.168.0.0       255.255.255.0;

reject ~192.168.0.1     255.255.255.255;

#

# Accept only outbound packets with a source address that is
within our

# class C network.

#

accept 192.168.0.0       255.255.255.0;

#----- Broadcast Addresses -----#
-----#

# Block all packets directed to the broadcast addresses within
our

# network

host 192.168.0.0         blockall;

host 192.168.0.255      blockall;

#----- Host Definitions -----#

# The Drawbridge host - UDP ports must be open for DNS queries

```

```

# Allow src=ftp-data and ssh; block ICMP echo request
host drawbridge.mybiz.comudpdefault, <1-65535/tcp out>,
                                <src=ftp-data/tcp in>, <ssh/tcp
in>,
                                <0-7/icmp in>, <9-18/icmp in>;

# No access in/out
host accounting.mybiz.com          blockall;
host transactserv.mybiz.com        blockall;

# Allow domain to hosts running dns for zone transfers
host dns.mybiz.com                 default, <domain in>;
host dns2.mybiz.com                default, <domain in>;

# NNTP host and phonebook server
host mailnews.mybiz.com            default, telftp,
                                <nntp in>, <time in>,
                                <csnet-ns in>, <domain in>,
                                <finger in>;

# Intranetapp server using port 4211
host intranetapp.mybiz.com          default, telftp, <4211/
tcp in>;

# Block www and gopher to internal server
host fileserver.mybiz.com          default, <!gopher in>, <!www in>;

# Telnet and FTP servers
host ftp.mybiz.com                 default, <ftp in>;
host intranet.mybiz.com            default, telftp;

#-----

```

Drawbridge has a command-line manager called `dbmgr`, which you can use to initialize the filter, start and stop it, load rule sets, monitor packet filter statistics, and set log criteria. This last option (logging) is particularly useful because you can set Drawbridge to report packets with any or all of the following criteria to the log file:

```

unknown event
initialized

```

```
incoming class D
outgoing class D
incoming port
outgoing port
incoming type
outgoing type
outgoing via accept table
incoming via reject table
outgoing via override table
incoming header too short
outgoing header too short
incoming D-O-S attack
outgoing D-O-S attack
incoming IP
outgoing IP
incoming fragment with IP offset == 1
outgoing fragment with IP offset == 1
incoming fragment
outgoing fragment
incoming MAC layer protocol
outgoing MAC layer protocol
```

The tcplogger, udplogger, and netwatch utilities each allow you to specify ports and protocols to watch for and log data in those areas to a text file. The netstat utility keeps network statistics and can be used to generate usage graphs and traffic patterns for your network.

Documentation

The Drawbridge comes with documentation but you can find additional information (as well as instructions on getting the package) at <http://drawbridge.tamu.edu/>.

Cost and Support

Drawbridge is distributed at no cost. Install it on as many servers as you feel like, and protect as many clients as you want. The biggest costs for using Drawbridge are the cost of the computer used to run it and the cost of the time spent by you or the administrator to configure and manage it.

Minimum platform requirements for FreeBSD with Drawbridge are easy to meet and should not be expensive:

- 486 processor or better
- At least two network interfaces
- 150MB of disk space
- 8MB RAM

OpenBSD and IpF

Another free operating system is rapidly becoming the security standard against which other operating systems are being compared (and unfavorably at that). OpenBSD was created by a group of security-minded BSD programmers who were unhappy with the deplorable frequency with which hackers had been finding holes in modern operating systems. Thus, they made their own version of BSD, which they proactively comb for security vulnerabilities and which they completely review whenever a new kind of security hole is found.

IpF for OpenBSD performs the same function (packet filtering and Network Address Translation) as IPChains or IPTables does for Linux and DrawBridge does for FreeBSD. If you want application proxies, you will have to install another service on top of Open-BSD—such as Squid or the TIS FWTK.

Major Feature Set

OpenBSD with ipf supports the following major features:

- Stateless packet-filtering rules are applied to every packet as it arrives, transits the OpenBSD routing stack, and exits.
- Proxies may be established by protocol-specific content filters provided by higher-level services such as TIS FWTK, Apache, or Squid.
- Network Address Translation (dynamic or static) is performed for packets transiting the routing stack to hidden, internal networks.
- DMZs can be established either by filtering access to an externally visible, protected subnetwork or by redirecting virtual public addresses to address-translated, protected hosts.
- VPN firewall-to-firewall and firewall-to-remote client options are provided using built-in IPsec functionality.
- Port redirection is natively provided by ipf.
- Clever use of ipf with Squid or FWTK can give you transparent proxies.
- OpenBSD with FWTK easily gives you reverse proxies (HTTP, SMTP, FTP, etc.) as well.
- Additional packages convert the regular OpenBSD Syslog reporting system to store logging information to databases and give you e-mail notification.

Minor Feature Set

OpenBSD with ipf supports the following minor features:

- Performance of OpenBSD packet firewalling is fast. Since it is integrated with the OpenBSD IP stack, the packet filter avoids the overhead of other firewalls that are implemented as user-level programs. This solution can easily handle a busy LAN connection to the Internet, even with Network Address Translation enabled.
- Command-line based configuration requires more skill in administration, but allows you to store policies in text files and use scripting tools for dynamic policy management.
- Remote management (using secure shell (ssh) or remote control software such as VNC)

allows you to manage your firewall from other computers in your LAN.

- The packet-filtering rules allow you to use NAT and forwarding for sockets to redirect traffic for particular services (such as HTTP, SMTP, and POP) to protected internal servers.

Security

OpenBSD has one of the most robust IP-level packet filters you can get. Not only does it protect against typical attacks such as source routing, malformed packets, and so on, but it also guards against more esoteric threats such as connection hijacking by TCP sequence number prediction. `ipf` gives you all the regular rule options, such as filtering on the SYN bit, source and destination IP addresses, source and destination ports, packet type, and most other TCP/IP header data elements. Network Address Translation is built into the packet filter, so you can use the same rules to specify the kinds of packets that will be translated and the kinds that will not.

Since OpenBSD does not inspect the data portions of the packets it manipulates, you will need a proxy server to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example). (See the sections on Squid and the TIS FWTK package.) Also, many web servers will also act as HTTP proxies, and you can use servers for store-and-forward protocols (such as SMTP and NNTP) without modification as protocol proxies for their services. Ideally, you will run these services on a computer other than your firewall, and you will use address translation to redirect the appropriate traffic to and from these servers.

The OpenBSD filtering software evaluates every packet received by the network adapters in the firewall computer, according to a set of rules you established when you installed the software. The rules are applied in order, one at a time, until the filter finds a rule that matches the packet and specifies a terminal action, such as `ACCEPT` or `DROP`. Since the rules are applied in order, it is vitally important to craft the rules in the right order.

Interface

OpenBSD packet filtering is administered using the `ipf` command and by editing the `ipf.rules` file, which contains the rules that the packet filter will use to protect your network. Figure 16.4 shows an example of configuring `ipf`.

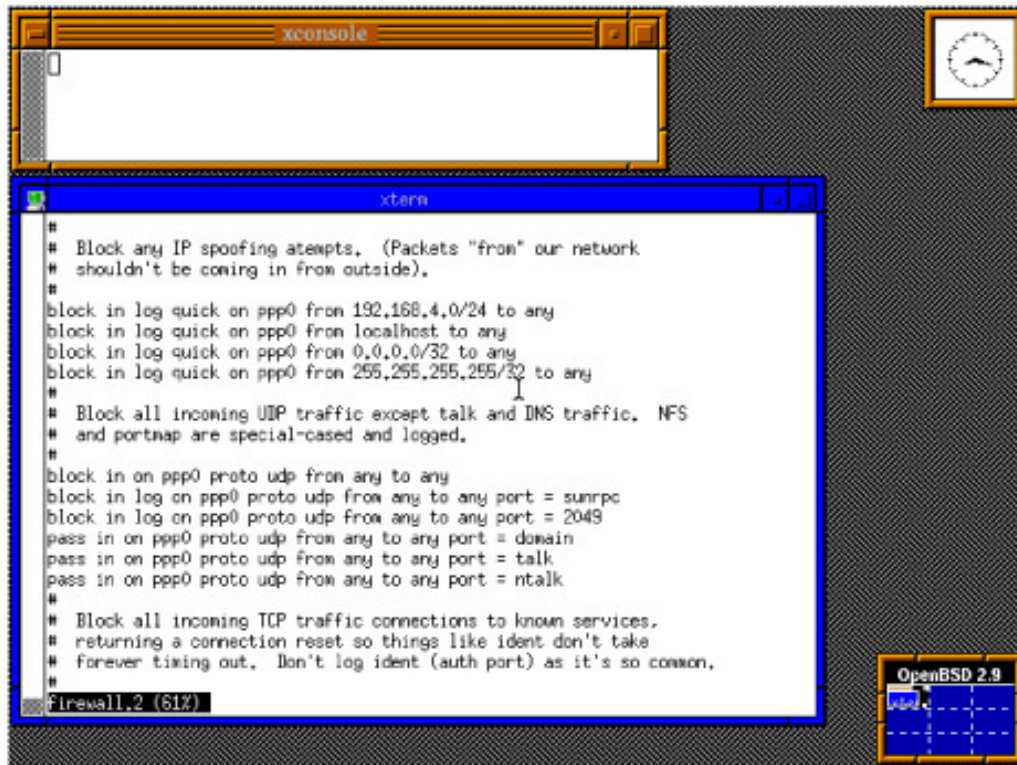


Figure 16.4: IPF takes its configuration from a text file.

Documentation

OpenBSD and ipf are documented in the typical Unix manner—readme and install files come with the software installation package (which may already be installed on your OpenBSD system) and documentation of those command line tools can be found from the man (manual) command.

Tip You can find step-by-step instructions for installing OpenBSD with ipf on the Internet; go to <http://www.openbsd.org/> or just search on OpenBSD and ipf from any search engine.

Cost and Support

You can download OpenBSD from <http://www.openbsd.org/> for free, or you can order the installation CD-ROMs for a modest price. The biggest costs for using OpenBSD are the cost of the computer used to run it and the cost of the time spent by you or the administrator to configure and manage it.

One great feature of OpenBSD is the online installation option that works entirely over the Internet. The process is simple: You download a boot floppy image, write it to an actual floppy, and use it to boot. The boot floppy asks for IP address information upon boot and then connects to the OpenBSD download site automatically, downloading and installing OpenBSD from the remote site. It's as fast as your download would have been anyway, completely up-to-date, and eliminates the CD-ROM/file shuffle that would normally take place. Other OS vendors should take a lesson from this very modern installation option.

Minimum platform requirements for OpenBSD are easy to meet and should not be expensive:

- A recent distribution of OpenBSD
- 486 processor or better, including any processor architecture for which a version of OpenBSD exists

- At least two network interfaces
- 200MB of disk space
- 32MB RAM

Packet Filtering with DOS and IPRoute

Some readers who have been using the Internet for a decade or more may be wondering why modern packet filters require a computer as powerful as a supercomputer used to be just to push packets around. After all, a Pentium computer is a fast processor, and even 8 megabytes of RAM is a lot compared to the computers of a decade ago, all of which operated on the Internet just fine. Many organizations have plenty of obsolete 286 and 386 computers lying around.

IPRoute is a packet filter and Network Address Translator that runs on DOS, allowing you to use those older, less powerful computers. IPRoute is not exactly free (there is a \$50 license fee), but then neither is DOS. They are both cheap enough to merit being a part of this chapter, because the money you will save by recycling your otherwise useless computers more than makes up the difference in cost.

Just because IPRoute runs on DOS doesn't mean it is limited in functionality. It is a fully functional packet filter, and NAT, and even provides dial-on-demand IP connections for home networks and workgroups that use modems to connect to the Internet. This functionality makes IPRoute especially useful for home offices and very small companies that don't have a constant IP connection.

Major Feature Set

IPRoute supports the following major features:

- Stateless Packet Filtering
- Packet-filtering rules are applied to every packet as it arrives, transits the IPRoute routing stack, and exits.
- Network Address Translation is performed for packets transiting the routing stack to hidden, internal networks.
- IPRoute will automatically establish dial-up connections for intermittent IP links.
- VPN between IPRoute installations is included in U.S.-only distributions of the software.
- IPRoute includes the ability to combine dial-up connections to increase available bandwidth.

Minor Feature Set

IPRoute supports the following minor features:

- Performance of IPRoute on top of DOS isn't the fastest, especially on an older and slower computer, but then IPRoute is most often used for slower Internet connections such as dial-up analog modems or ISDN anyway. IPRoute is entirely sufficient for connection speeds less than 1Mbps.
- Command-line based configuration requires more skill in administration but allows you to store policies in text files and use scripting tools for dynamic policy management.
- Address Translation and forwarding for sockets is performed using the IPRoute rules allowing you to redirect traffic for particular services (such as HTTP, SMTP, and POP) to protected internal servers.

Security

Since DOS doesn't have a built-in IP protocol stack, IPRoute implements one, as well as the packet-filtering and NAT functions. All packets are processed by the packet filter before being routed or translated. IPRoute provides the full range of packet-filtering options, including the SYN bit, source and destination IP addresses, source and destination ports, packet type, and so on.

Because IPRoute does not inspect the data portions of the packets, and because DOS is a single-tasking operating system, you will need a proxy server running on another computer to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example). Many web servers will also act as HTTP proxies, and you can use servers for store-and-forward protocols (such as SMTP and NNTP) unmodified as protocol proxies for their services. Ideally you should use address translation to redirect the appropriate traffic to and from these servers.

IPRoute evaluates every packet received by the network adapters in the firewall computer according to a set of rules stored in its database in RAM (loaded when IPRoute starts). The rules are applied in order, one at a time, until IPRoute finds a rule that matches the packet and specifies a terminal action, such as ACCEPT or DROP. Since the rules are applied in order, it is vitally important to craft the rules in the right order.

Interface

Like the other free firewalls discussed in this chapter, IPRoute is configured using command-line scripts. One script starts the service and establishes the Telnet command interpreter (so you can administer the firewall remotely), and other scripts manage such functions as dialing the ISP and establishing the packet filtering and Network Address Translation rules. The Typical IPRoute Configuration Scripts sidebar illustrates these rules.

Typical IPRoute Configuration Scripts

```
. IPR script for PPP using a static IP address

; We have an ethernet adapter (and configured packet
; driver) for the local LAN.

packet en0 0x60 192.168.0.1/24

; We connect to the Internet using a serial port modem

async sl0 0x3f8 4 57600 do_sl0

; We make it the default route

route * sl0 perm

; use RIP on the local LAN.

rip en0

; Start up the command interpreter

command

exit
```



```

. Filter and NAT configuration script

do_sl0:

set trace on

;

;   Configure packet filtering.
;

;   Pass through connection attempts from the outside
;   only if they come from the outside.

filter permit in tcp-xsyn !192.168.0.0/22 24.0.148.200

;   Drop and log attempts to spoof internal addresses, or our
external address

filter log drop in * 192.168.0.0/22 *

filter log drop in * 24.0.148.200 *

;   Allow UDP responses.  NAT will keep out unsolicited pack-
ets.

filter permit in udp * 24.0.148.200

;   Allow incoming mail and FTP connections

filter permit in tcp-syn * 24.0.148.200:25

filter permit in tcp-syn * 24.0.148.200:21

filter permit in tcp-syn *:20 24.0.148.200

;   Drop and log everything else.

filter log drop in * * *

;

;   Configure Network Address Translation

;

;   Redirect SMTP mail to our mail server.

nat tcp 192.168.0.2:25 24.0.148.200:25

;   Redirect FTP to our FTP server.

nat tcp 192.168.0.3:21 24.0.148.200:21

nat tcp 192.168.0.3:20 24.0.148.200:20

;   Let outgoing connection requests go out.

```

```

nat * * 24.0.148.200

;

;   Configure modem control and set up for demand dial.
;

set interface rts flow

hangup:

set interface dtr off

delay .5

set interface dtr on

on packet dialout

wait 0

;

;   Dial and establish a PPP connection.

;   Hang up after 10 minutes of inactivity.

;

dialout:

send "ATDT 555-1212\r"

wait 1

on timeout hangup

wait 45 "CONNECT"

;   Read the rest of the CONNECT message into symbol "speed".

on cd lineup

read 5 "\r" speed

wait 60

lineup:on cdloss hangup

on timeout hangup

log "Connected at $speed"

set ppp login somebody somethingsecret

ppp 600 24.0.148.200/24

goto hangup

```

Documentation

The IPRoute software comes with documentation, but you can find additional information (as well as instructions on getting the package) at the IPRoute website: www.mischler.com/iproute/.

Cost and Support

IPRoute is commercial software; you have to pay a \$50 license to run it. You can down-load a 30-day, time-limited evaluation copy to try it out, however. Technically, you must also have a licensed copy of MS-DOS, but IPRoute should run just fine on FreeDOS.

Minimum platform requirements for IPRoute can be met by almost any IBM compatible computer that you can actually get to boot:

- 286 processor or better
- At least two network interfaces, with DOS packet drivers
- A floppy or hard drive (capacity unimportant)

Case Study: Free vs. Commercial Firewalls

Are you looking for freedom from licensing costs, or are you more concerned with freedom from configuration hassles? Do you want to be free to fix the bugs in the software yourself, or is it more important that you be able to require the vendor to fix the security hole? The firewalls reviewed in this chapter are not without cost—even monetary cost—because the time it takes you to set them up has value too (presumably).

That said, a system administrator well versed in Unix lore and proficient with scripting tools may in fact be better able to tailor these firewalls to your network's needs than he or she would be able to make Firewall-1 or a handful of SonicWALLs do the job. The tool must fit the job, and the tool must fit the tool user.

We use both free and proprietary firewalls in networks we manage. For the majority of our medium-sized customers, we have dedicated firewalls connecting their LANs to the Internet. These devices are easy to set up, require little maintenance, and have all the features our customers need.

Our own requirements are a little more so-phisticated, however, because we must protect our own servers, maintain IPSec connections to our clients' networks for monitoring and rapid incident response, and have access back into our own network from roaming laptops wherever we may be, even behind a NAT. Oftentimes, we have to use one VPN connection from a laptop to get to where we can use another VPN connection to remotely fix a client problem, and VPN client software is notoriously unfriendly to other VPN software running on the same computer. For that reason, we have a heavy-duty commercial firewall protecting our LAN and maintaining VPN connections to our clients (isolating the clients from each other), while we VPN into (DMZ-isolated) free-software based servers for remote and bounce-through access. This mixed bag of technology suits us, because we have to understand the systems anyway.

Part V: Commercial Firewalls

Chapter List

Chapter 17: Windows Firewalls

Chapter 18: Unix Firewalls

Chapter 19: Device and Specialty Firewalls

Part Overview

Featuring:

- Feature comparison of firewalls
- Technology details of firewalls
- Recommended firewalls for various platforms
- Commercial firewalls for Windows
- Commercial firewalls for Unix
- Commercial firewalls for Open–Source operating systems
- Commercial firewalls for OS/2
- Commercial firewalls for NetWare
- Dedicated firewalls that include their own operating systems

Chapter 17: Windows Firewalls

Overview

There are now so many firewall products available that it's difficult to determine what you should use. This chapter will familiarize you with the firewall market, applying the theoretical information in the first part of this book to the practical selection of a real firewall. This chapter is specific to firewalls specific to Windows, although most of them also have versions that run under other operating systems as well.

This chapter details the following firewalls:

- Checkpoint Firewall-1
- NetGuard Guardian NCC
- Symantec Enterprise Firewall
- Microsoft Internet Security and Acceleration (ISA) Server

These firewalls represent the high-end firewall market for firewalls that run on the Windows NT kernel-based operating systems. These firewalls make use of the user interface, services architecture, and in some cases the network interface functionality of Windows, and add only those components related directly to security of the operating system. Basing the firewall on an existing operating system is a double-edged sword. It allows the security systems vendor to concentrate on writing security software rather than operating system software, but it can also make the resulting product vulnerable to flaws in the operating system if the vendor hasn't taken special preventative precautions. With the solitary exception of Microsoft ISA Server, the firewalls profiled in this chapter support a remarkably similar set of technologies. They all cost about the same amount of money and are ICSA certified.

Note ICSA is a commercial organization that tests firewalls and security devices. The vendor whose equipment is tested pays them. While they are an independent third party who affirms the security of a device, they don't back that affirmation up with any sort of guarantee. And they've certified one device that I've hacked through myself because of its security flaws—the Avaya VPNet router, which leaks SNMP information about the interior of the network and which until recently would forward source-routed packets through to the interior of the network. So, while their certification is more than just a vendor saying it's secure, it's not much more than that.

The group is divided into two types based on their primary security posture:

- **Stateful Inspection Filters** Use complex filters based on retained information about connection state and protocols to either block or pass traffic. Firewall-1 and Guardian fall into this group.
- **Proxy Servers** Receive and then completely regenerate allowed services through the gateway, and ignore protocols for which there is no established proxy. Gauntlet, Symantec Enterprise Firewall, and ISA Server fall into this group.

The primary security posture of a firewall doesn't tell the whole story; most stateful inspectors include proxy or proxy-like services, and most proxy servers include stateful packet filters. The division in this case depends upon which philosophy the architecture of the firewall is based, and which services are added on to shore up deficiencies in the basic architecture.

There are two things you will not find in this chapter:

- Performance ratings
- Hacking tests

We decided not to include performance information in this chapter because we believe that performance should not be a deciding factor in your security posture. This would be something like comparing the top speeds of tractors—performance isn't the point. The essential problem is that more inspection and rigor takes more time, so the better a firewall is, the slower it will perform. If you are in the rare circumstance that you must use a high-performing firewall, use a stateful inspector. Otherwise, proxy servers provide more security, albeit at considerably reduced performance rates.

We performed a number of hacking tests against these products once they were properly secured using publicly available hacking tools. We were not able to find any case in which a firewall was susceptible to intrusion or denial of service except when we knew architectural flaws existed in the software. So we decided not to write about our lack of results. Psychological attacks using forged e-mail or rogue websites remain the only ways we know of to penetrate these firewalls.

Note This chapter contains discussions of various firewall products based upon evaluation software and hardware provided by the vendors, their documentation, and our installation and testing of the evaluation software or device (except where noted). We were not able to review the product source code for dormant flaws and cannot ensure that these products will remain secure in a continually changing security environment.

Each firewall is detailed in its own section throughout the remainder of this chapter.

Checkpoint Firewall-1

Checkpoint Firewall-1 is a policy-based stateful inspection filter with an integrated Network Address Translator and a small set of nonintegrated protocol-specific security filters for common Internet protocols. Checkpoint Firewall-1 was the best-selling firewall in the world until about two years ago, when the firewall device market overtook it.

The most recent version of Firewall-1 (included as part of the Checkpoint NG Enterprise Suite or available as a separate product) is an incremental improvement that removes most of the most annoying configuration problems with Firewall-1: the need for routing rules on the host OS to support Network Address Translation, the requirement for proxy-ARP, and so forth. By addressing these issues, Firewall-1 is now substantially easier to successfully install and configure than prior versions had been.

Checkpoint developed the concept of stateful inspection to improve the security of packet filters without requiring the overhead of proxy servers. Once a packet passes the suite of tests applied by the inspector module, the original packet is forwarded into the network. This means that any deformations not detected by the inspector module are passed through without modification by the firewall module.

Stateful inspection is a middle ground between simple packet filters and application proxies. Because stateful inspectors maintain state information about each connection, they can make more rigorous pass/fail checks on packets. But they do not usually have the ability to monitor the internal content of the various protocols, so they are more closely related to packet filters than to proxy servers.

Firewall-1 solves this problem to some degree by allowing plug-in protocol filters that are similar to actual proxies. These protocol filters understand the content of popular protocols like HTTP, SMTP, and FTP (the three provided with Firewall-1), so they can inspect and make pass/fail decisions on those protocols. These filters are able to perform high-level filtering functions like Java blocking and attachment stripping. Filters remain less secure than proxies because the packets are routed through the firewall, rather than being re-created as they are in proxies. The Firewall-1 SMTP filter is a true proxy, as it writes e-mail to its disk and then has a separate service forward the e-mail through the gateway. This method is designed to prevent the buffer overflow problems that plague e-mail systems.

Don't confuse content filters with simple protocol support. Firewall-1's documentation claims to support over 120 protocols out of the box. By "support," they mean they've defined an object that encapsulates a protocol's protocol header number, not a content filter. Firewall-1 provides content filters for just three common protocols: HTTP, FTP, and SMTP.

The management console requires a Win32 or Sun Solaris host. The firewall modules can run on Unix, Linux, or Windows computers or on numerous commercial routers from Cisco, Bay Networks, and others. Perhaps the coolest design feature of Firewall-1 is that with it you can convert your existing inventory of border routers into strong firewalls.

Firewall-1's documentation assumes you have a working knowledge of TCP/IP and the platform upon which you are installing the software.

Major Feature Set

Firewall-1 supports the following major features:

- Stateful packet filter
- Protocol-specific transparent proxies (HTTP, SMTP, and FTP)
- Reverse proxies (HTTP, SMTP, FTP)
- Network Address Translation
- Port redirection
- DMZ support
- VPN firewall-to-firewall and firewall-to-remote client add-on components (available at additional cost)
- VPN Client Software (Windows 98/NT/2000/XP, Macintosh, Unix, Linux)
- Logging and e-mail notification (including SNMP trap)

VPN Features The VPN-1 component of the Checkpoint NG Suite is actually a separate product from Firewall-1, but they are closely integrated. The VPN-1 product supports a wide range of remote client authentication schemes, including Checkpoint's proprietary authentication, NT Challenge/Response, SecureID, RADIUS, Acent Pathways Defender, TACACS/+, and X.509 certificates.

For encryption, VPN-1 supports AES, DES, and 3DES using IPSec/IKE and supports IP Compression (IPCOMP). VPN-1 can be integrated into a Public Key Infrastructure (PKI) system.

The SecuRemote VPN client for VPN-1 is available for Windows 98, NT, 2000, and XP, as well as for the Macintosh, Linux, and most major variants of Unix.

Additional Optional Features Checkpoint NG Enterprise Suite includes the Flood-Gate

Quality-of-Service traffic shaper, the ConnectControl module for load balancing and high availability, and separate firewall high availability features.

Minor Feature Set

Firewall-1 supports the following minor features:

- Content filtering (Java, ActiveX, Virus scanning, URL blocking)
- Scan detection, spoofing detection, and automatic blocking
- SYN flood protection
- Security Server (Proxy authentication)
- Real-time monitoring
- Centralized administration
- Policy-based configuration and management
- Highly configurable alerting
- Execution of arbitrary programs upon event detection
- Diagnostic tools
- DHCP, SNMP
- LDAP integration
- Demand dialing of PPP connections
- Split DNS
- Support for third-party content scanners
- SQL proxying
- Online Help
- Host OS Boot security
- Process Watchdog

Policy-based configuration and management Makes it easy to view, manage, and understand the configuration of the firewall. Like most GUI-based firewalls, Firewall-1 lets you create protocol definitions called objects that associate a friendly name with a collection of protocol identifiers like the port number and IP protocol type. This way, you can work with objects like FTP instead of TCP Port 21, so you won't get confused during the configuration process. Because the abstraction allowed by identifying protocols, addresses, users, and time ranges as named objects is easy to understand, management is simple. This tends to reduce the number of mistakes made when configuring the firewall.

Centralized client/server management What separates Firewall-1 from the majority of policy-based firewalls is its support for a single enterprise-wide policy interface which can automatically generate custom-compiled firewall policies for each firewall in your enterprise. So, rather than managing firewalls separately and mapping your security needs to each firewall's specific place in your network, you can maintain and configure a single policy that is automatically customized for each of your firewalled devices.

Content Vectoring Protocol Allows you to plug in filters to handle very specific protocols like HTTP, mail, and FTP. CVP-compatible filters can strip attachments and executable content, perform virus checking, URL blocking, or any other protocol-specific filtration. NAI and Symantec both make CVP-compatible virus scanners that work with Firewall-1 and are available at additional cost.

Automatic address translation Handles objects on an individual basis. Once an object is defined and an address translation mode is assigned to it, address translation rules will be automatically generated for every case where the object is used in the rule base. Address

translation rules can be manually created for those cases where automatic translation doesn't accomplish your goals.

Firewall module synchronization Allows firewalls to trade state with each other. If two firewalls on the same connection are used, one can fail without affecting the connections running through them. This feature can also be used to perform load balancing across a range of firewalls.

Interface

Firewall-1 is a client/server architecture that allows you to centrally control any number of firewall modules from a single management console. The GUI is easy to read and comprehend without being overly busy. The latest version has solved a number of the annoying design problems that plagued earlier versions of Firewall-1 to create a simple and very usable policy editing interface.

Firewall-1 abstracts devices, users, and networks as objects defined by IP or network addresses and referred to by a uniquely assigned name identifier. Pass/drop rules are defined by selecting a source object (including "Any" to encompass the Internet as well as internal systems), a destination object, one or more protocols, the action to apply, and the logging or alerting level. The collection of rules is called a rule base; it is synonymous with the strategies used by most firewalls. Figure 17.1 shows the Firewall-1 interface with a complete rule base showing.

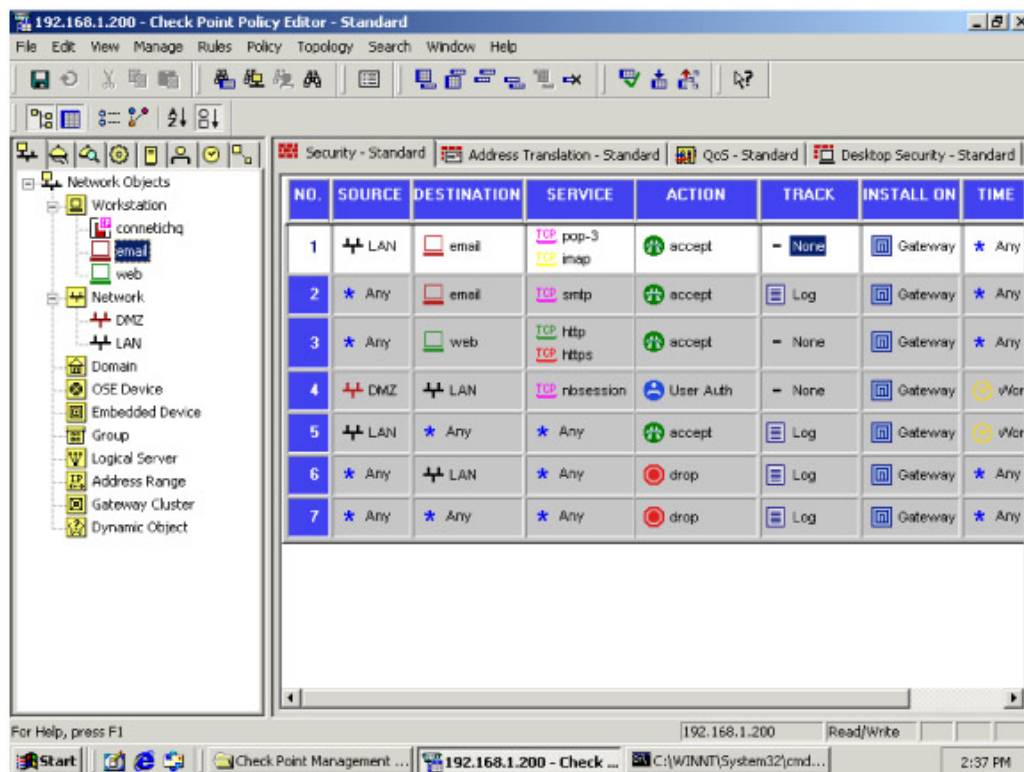


Figure 17.1: Firewall-1's rule-based interface

Rules are interpreted from the top to the bottom of the rule base as it is displayed on the screen. The first rule that applies to a packet is used, so a number of rules pertaining to the same protocol can be added in a very intuitive and obvious manner. This allows you to create various levels of security for different groups. The last rule in the rule base is "Any source, any destination, any protocol: drop with no logging." This rule is implicit and is not shown in the rule base, but it guarantees that anything not specifically allowed is specifically denied.

Network address directives are assigned per object, so once the rule base is complete, a NAT strategy is automatically defined. You can add manual address translation rules, but that is usually not necessary.

Once a policy is defined, it must be compiled and applied to the appropriate gateways. This is easy to complete; but unfortunately it is possible for the GUI to allow you to create policies that won't correctly compile. In that case, you must go through something of a compile/debug cycle to create a working policy. A solid user interface would simply prevent you from creating problem policies in the first place.

Security

Checkpoint devised the idea of the stateful inspection packet filter, which improves on the basic packet filter by more closely examining the packets used to set up connections and store connection information (the state). This stored state is used to determine which packets should be passed and which should be dropped based on their participation in a connection. Stateful inspection is very fast because the computation done to examine packets is fairly slim, and once a connection has been established, the filtering of packets through the connection takes very little time. Stateful inspection filters are capable of operating nearly as fast as a standard IP router.

But throughput is not the purpose of a firewall; it is merely a feature. Strong security, which can only be achieved through rigorous examination of all possible protocol information, remains paramount. Because stateful inspectors like Firewall-1 perform only cursory examination of TCP-layer information and do not typically filter the contents of packets, they are not as secure as pure proxy servers like NAI Gauntlet or Symantec Enterprise Firewall.

To close that security boundary, Firewall-1 includes a small set of security filters for common services like SMTP and HTTP; these filters are not well integrated into the management paradigm, however. Firewall-1 also includes a protocol filter for HTTP that is capable of stripping out dangerous content like executable files and Java applets.

Documentation

Firewall-1's online documentation is among the best in the business. It teaches firewall theory, application, user interface, and is packed with examples. It is professionally written and appropriate for the target audience. Most network administrators will be able to establish a firewall without technical assistance now that most of the major problems in the Firewall-1 configuration set have been eliminated.

If you intend to purchase and install a firewall by yourself without prior experience, you should consider Firewall-1 based on the strength of its documentation. Any Unix Administrator, Microsoft Certified Systems Engineer (MCSE), or equivalent should be able to figure out Firewall-1 from the documentation alone and construct a reasonable, secure firewall policy for it.

Cost and Support

Firewall-1 is sold a number of different ways:

- Single gateway products support a specific number of users. The management console and gateway are installed on a single machine. This product is sufficient for small businesses with fewer than that number of IP addresses on their network (which is how the firewall determines how many hosts it will work with).

- Enterprise products protect an unlimited number of internal hosts and are sold on a per-module basis. You purchase the number of firewall modules you require (one per border gateway) and the number of encryption modules you require to support the VPN functionality.

Minimum platform requirements for Firewall-1 are easy to meet and should not be expensive:

- Pentium processor
- At least two network interfaces
- 40MB of disk space
- 128MB RAM
- CD-ROM drive

Checkpoint is stingy with online support and charges an exorbitant \$400 per incident for telephone technical support (for which they will not guarantee a solution to your problem). I can understand not wanting to deal with first-time network integrators, but it seems that Checkpoint has decided that technical support is a lucrative market. Competition will inevitably change their minds. That said, their technicians seem very competent, as far as I could determine without providing my credit card number.

Firewall-1 is sold on a per module basis, with a simple 5-user Small Office product for \$300 and a 250-user module costing \$6000. You can get modules for firewalling, VPN, and Quality-of-Service at various comparable prices, and you need a module for each device in your enterprise that you want to firewall. Operating systems supported are Windows, Linux, and Solaris.

The VPN-1 and remote authentication module are about as expensive, and costs for additional users hover around the \$100-per-user point.

Symantec Enterprise Firewall

Symantec Enterprise Firewall firewall (formerly Axent Raptor, formerly Raptor Eagle Firewall) is Gauntlet's strongest competitor in the area of security. Like Gauntlet, Symantec Enterprise Firewall is a security proxy. Unlike Gauntlet, Symantec Enterprise Firewall does not include the adaptive proxy filter technology that increases the speed of Gauntlet to near that of a stateful inspector. Symantec Enterprise Firewall is among the fastest proxy firewalls, however, and is capable of handling dedicated circuits up to T3 (45Mbps).

Symantec Enterprise Firewall runs on Windows NT 4 SP 6a, Windows 2000, and Sun Solaris (SPARC), and is multithreaded to take advantage of multiple processors. Symantec Enterprise Firewall can be used with Windows NT Cluster Server and Windows 2000 Advanced or Datacenter Server to create high-availability firewall services. However, Symantec Enterprise Firewall is not compatible with Windows 2000's Advanced Server's Windows Load Balancing service.

Unlike most firewalls covered in this section, Symantec Enterprise Firewall relies upon "best-fit" policies that are not order-dependent. This means that the firewall applies the policy that most closely applies to each connection, rather than filtering the connection down through a policy rule base until either a pass condition is met or the connection is dropped.

System requirements are as follows:

- Windows NT 4 SP 6a Server or Windows 2000 Server SP-1

- Intel Pentium II 233 (Because Symantec Enterprise Firewall is a proxy server, it is compute bound, so you should use the fastest available processor.)
- 128MB RAM
- 2GB Disk
- Two network interfaces

Major Feature Set

- Packet Security Filter for the gateway
- Network Address Translation
- Security proxy
- Remote Authentication
- VPN support is provided through the add-on Symantec Enterprise Firewall VPN and Symantec Enterprise Firewall Mobile VPN products.

Packet Filtering

Unlike other firewalls, Symantec Enterprise Firewall does not allow Network-level routing and therefore does not include a packet filter. All data, even low-level information like ICMP and TCP generic services, are routed through Application-layer proxy services and regenerated on the firewall. This is the most secure method of passing information between interfaces, as it guarantees that no malformed packets can cross through the gateway.

In addition to performing no routing, the firewall automatically drops source-routed packets and packets containing internal addresses that appear on external interfaces. These packets are dropped before any connection proxying can be performed on them.

It is not entirely clear whether the firewall is capable of protecting the operating system's TCP/IP stack from denial-of-service attacks because it does not appear to include an NDIS-layer adapter driver. Considering that the installation requires updating the operating system to the latest security hotfixes, it's likely that Symantec Enterprise Firewall is indeed susceptible to network-level attacks directed at the operating system. None of these attacks provide access to the system, but they can deny Internet services.

Network Address Translation (NAT)

Symantec Enterprise Firewall relies primarily on its proxy service to perform the standard many-to-one address translation. But it also uses reverse address translation to support services on interior machines and true Network Address Translation through a feature Symantec Enterprise Firewall calls Virtual Clients. The Virtual Clients facility also allows support for Illegal Network Address Translation.

Security Proxies

Symantec Enterprise Firewall is primarily a security proxy that uses separate security proxies for every supported protocol. Third-party products must be used to perform virus scanning and Java filtration. Symantec Enterprise Firewall includes security proxies for the following services:

- SMB/CIFS (Windows/LAN Manager network file and print sharing)
- SQL*Net (Oracle SQL servers)
- Telnet
- FTP

- SMTP
- HTTP 1.1
- HTTP-FTP
- HTTP-Gopher and Gopher+
- HTTPS
- H.323
- Ping
- NNTP
- RealAudio and RealVideo
- NDS
- NTP (Network Time Protocol)

Authentication Support

Symantec Enterprise Firewall can be configured to support the following authentication protocols:

- Security Dynamics ACE
- BellCore S/Key
- Defender (by Axent)
- CRYPTOCARD
- Gateway password
- Windows NT Challenge/Response
- RADIUS
- TACACS+

Minor Feature Set

Symantec Enterprise Firewall includes support for the following minor features:

- **MIMESweeper virus scanning** This feature can be used to strip viruses out of downloads and attachments. Symantec Enterprise Firewall is missing support for the standard CVP content-vectoring protocol, however.
- **URL blocking** This feature is based on a client/server updated list of sites that have been categorized. There's no real way to keep up with the ever-changing world of the unseemly, however, so I doubt that any simple URL filter would actually keep people from accessing this sort of content.
- **Paging and audible alerts** This feature can be used if your firewall has a Hayes-compatible modem and/or a sound card. The paging alert is especially useful for administrators who want to maintain a real-time response capability.
- **Transparency** This feature is supported by Symantec Enterprise Firewall. You won't have to configure client applications or rely upon clients that are proxy compatible to use Symantec Enterprise Firewall.
- **Illegal NAT** This support feature, using the Virtual Clients facility, allows you to perform client address translation through the gateway for networks that use illegal IP addresses.
- **Dual DNS** This configuration feature allows different DNS names to be served to the public and private sides of the proxy.

Security

Symantec Enterprise Firewall's gateway security architecture is extremely strong; it's highly unlikely that attacks through the firewall would succeed due to the proxy-only architecture. The application-layer support for Network Address Translation is also very strong and transparent.

Symantec Enterprise Firewall's Achilles' heel is its reliance upon a stable operating system and TCP/IP stack. Telling requirements in the Symantec Enterprise Firewall installation documents (like the necessity for the latest service pack) show that everything in the firewall operates above the Network layer. There appears to be no MAC-layer protection (such as a packet filter) for the operating system itself, so there's no support for things like anti-spoofing. This is a fairly common problem for pure proxies. Ultimately, this means that hackers could be able to bring your firewall down and cause a denial of service, but they would not be able to penetrate the firewall to access your secured network.

Symantec Enterprise Firewall does not include any native high-availability or load-balancing features and is not compatible with the Windows Load Balancing Service. However, you can use a RadWare plug-in to achieve high availability or load balancing; more information about this is available from Symantec.

Interface

Symantec Enterprise Firewall version 6 uses the Microsoft Management Console to achieve a highly integrated and very useful user interface—it's the best user interface I've seen on a firewall. Despite the recent purchase and name change, Symantec Enterprise Firewall still calls its MMC snap-in the Raptor Management Console, or RMC. The RMC is client/server based and can support any number of firewalls. The interface is hierarchical following the architectural requirements of the MMC. Figure 17.2 shows the user interface for Symantec Enterprise Firewall.

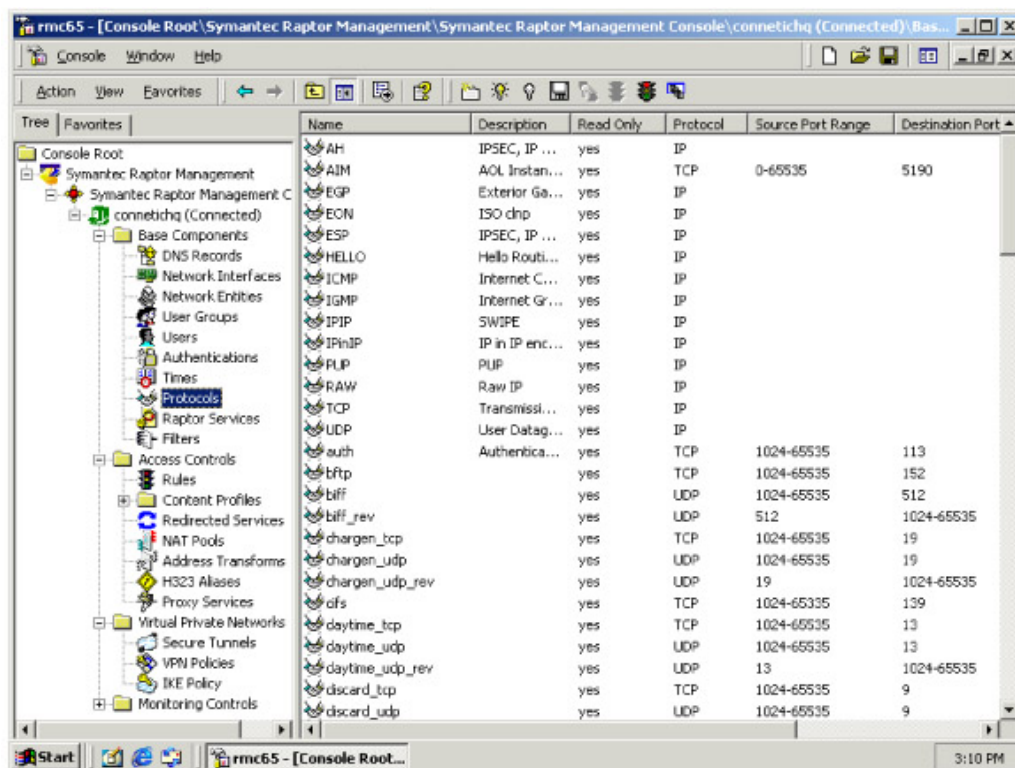


Figure 17.2: Raptor Management Console for the Symantec Enterprise Firewall user interface Management objects are completely hierarchical and very coherent, although the management interface is more complex than most firewalls. Network administrators familiar with the MMC should have no problems.

Documentation

The included documentation is very thorough and is task oriented rather than technology/training oriented, although solid coverage of basic topics is provided. You can easily download a trial version from Symantec's website, but obtaining pricing information is nearly impossible.

Cost and Support

Support is via the website or support agreement only.

Licensing costs for the firewall (with VPN)

- 1–100 users: \$4,000
- 1–250 users: \$8,500
- Unlimited users: \$12,500

There is also a "deviceafied" version that comes preinstalled on a Sun Raq 1U Linux server. Its prices are oddly about the same as the software-only versions, so it's a better deal—considering that you don't have to license an operating system or buy a computer.

Standard support

- 25 users: \$375
- 100 users: \$600
- 250 users: \$1,275
- Unlimited: \$1,875 per year

24 × 7 (no relation to this book) priority support is purchased as a single product rather than on a per-incident basis. Prices vary, depending upon which product modules you have. Prices without VPN support are:

- 100 users: \$900
- 250 users: \$2,000
- Unlimited: \$2,800

Microsoft Internet Security and Acceleration Server

Microsoft ISA Server replaces Microsoft Proxy Server 2.0 as Microsoft's firewall product. The two products have essentially no similarity whatsoever—if they share any code, it's not apparent. Proxy Server was a simple HTTP proxy server that Microsoft claimed was a firewall. ISA Server is a real ICSA-certified firewall that performs all manner of forward, reverse, and transparent proxying for numerous protocols.

The difference in maturity between the two products is so extreme that it's difficult to believe that they came from the same vendor within two years of each other. Unlike most of the other updated firewalls in this book, which underwent minor usability updates, name changes, and interface improvements, ISA Server is a completely new product that dramatically improves Microsoft's standing in the firewall market.

That said, ISA Server suffers from some problems. It relies heavily (though not completely) on

Windows 2000's built-in security mechanisms—those low-level (Layer 4 and below) problems that affect Windows 2000 are highly likely to affect ISA Server.

ISA Server's tree-browser interface, while easy to figure out and configure initially, does not have any sort of "overview" of your security policy the way that most policy-based firewalls do. Considering that there is usually more than one way to achieve the same result in ISA Server, it can be excruciatingly difficult to determine exactly what to change down the road when you want to reconfigure your policy.

Major Feature Set

Microsoft Internet Security and Acceleration Server provides the following major features:

- Stateful packet inspection
- Circuit layer (SOCKS) proxy
- Application-layer filtering in-and-outbound
- IPSec, PPTP, and L2TP VPN
- Numerous authentication methods

Stateful packet inspection ISA Server builds on the built-in filtering in Windows 2000 by adding a stateful inspection filter (Windows 2000's packet filtering is stateless). Stateful inspection ensures that inbound return channels on the firewall are closed when the TCP session had ended or timed out. The stateful inspection facility is also used to provide the SecureNAT feature of the firewall, which improves upon Windows 2000's built-in network address translation by providing circuit layer regeneration rather than simple packet address translation.

Circuit Layer Proxy ISA Server allows you to define circuit layer proxies that are protocol generic and which will work with any Windows SOCKS-based protocol, as well as with most Berkeley sockets-based protocols. A circuit layer proxy is a generic proxy that acts as the end point for TCP sessions and then regenerates them on the other side of the firewall. The effect is much the same as a stateful inspection firewall, except that the original TCP/IP packets themselves are not passed through—they are regenerated on the outbound interface so that no Network- or Session-layer malformations make it through. A circuit layer proxy is essentially the same thing as setting up a Unix redir service to receive and redirect a socket on a Unix machine.

Application-layer filtering ISA Server comes with a number of application layer security filters that can process rules for a specific application. The default set of filters includes:

- ◆ SMTP
- ◆ FTP
- ◆ HTTP

Numerous third-party content filters, virus scanners, and content blockers are available now for ISA server. If you have programming talent available, you can use the SDK to create your own content filters as well.

Virtual Private Networking ISA Server manages the built-in IPSec, PPTP, and L2TP VPN services of Windows 2000. Although it does not provide these services, it does create a simple, unified management platform and provides wizards for setting up such common schemes as a VPN remote access server and firewall-to-firewall VPN connectivity.

Authentication Authentication is another area where ISA Server gets a free feature from its strong integration with Windows 2000. Because it passes authentication through to the server below, ISA Server can support all Windows 2000–supported authentication schemes, including plain–text passwords, NTLM encrypted passwords, Kerberos, X.509 certificates, smart cards, and third–party biometrics.

But with the good comes the bad—many of these authentication schemes are weak, and using them for the sake of convenience can seriously jeopardize your overall security posture. Although Microsoft considers it a "feature" that ISA Server can be configured to authenticate users based upon their internal account names and passwords, I would never use internal accounts for external security because usernames and passwords are far too easy to obtain from other sources, like malicious websites and Windows 95/98/Me .pwd files. Always use certificate–based authentication at the very minimum when configuring ISA Server as a remote access server.

Minor Feature Set

Microsoft Internet Security and Acceleration Server has the following minor features:

- Strong intrusion detection
- OS system hardening
- Highly configurable logging
- Ability to send e–mail or run arbitrary programs in response to alerts
- Transparent proxying
- Inbound content caching
- Outbound published content caching
- Policy–based configuration
- SSL bridging
- High availability and load balancing
- Quality of Service bandwidth allocation
- Enterprise management

Strong intrusion detection ISA Server's intrusion detection filters are based on Internet Security Systems' ISS intrusion detection suite. While I'm not a huge fan of fingerprint–based ID systems running stand–alone, they're a marvelous addition to a firewall when your only other option would be the simple "someone connected to this blocked port" logging that you get with most firewalls. The intrusion–detection filters in ISA Server work at the Network (packet) layer and at the Application layer. Third–party Application–layer filters can trigger intrusion–detection alerts based on malformations in the protocols they filter.

OS system hardening ISA Server performs automatic operating system hardening for three different purposes as part of its setup wizard. This OS hardening feature was sorely lacking in Proxy Server 1 and 2, because Microsoft naively assumed that everyone knew how to make an NT or 2000 box secure. By packaging OS hardening as a "select one" configuration, ISA Server closes the holes that made Proxy Server a mere security placebo.

The OS Hardening Wizard allows you to select "dedicated mode" for servers that are dedicated to firewalling, or "secure mode" for installing on public Internet web and database servers that must protect themselves. There's also a hybrid mode that combines the features of the two.

Strong logging and alerting Like all truly good firewalls, ISA Server gives you plenty of

options when an alert comes in. You can send an e-mail, log it to the Windows event log, or stop or start a specific service. Stopping services automatically would be useful, for example, when your web server comes under concerted attack from the Internet and you're more interested in preventing intrusion than in remaining online. Finally, ISA Server can launch and run arbitrary programs in response to specific alerts, which gives you complete flexibility in dealing with intrusion attempts.

Transparent proxying ISA Server supports transparent proxying and Application-layer filtering for clients running in SecureNAT mode on the interior of the network. This eliminates machine-by-machine proxy configuration, which was required in MS-Proxy Server and was relatively easy for users to defeat.

Inbound and outbound caching ISA Server's ascent from a simple HTTP proxy leaves it with strong HTTP caching abilities. You can configure the server to cache content for inbound users, and even distribute the cache among ISA Servers in your enterprise. This can dramatically improve performance for websites that are frequently visited, which is actually becoming more common as businesses begin transacting with one another over the Web, but is ineffective for random surfing.

You can also configure an ISA Server (or an array of machines running ISA Server) to cache outbound content from your web server farm. So, for example, instead of having ten web servers that you have to maintain identical configurations of, you could have a single configured web server and an array of ten ISA Servers in front of it delivering its cached content. Of course, your mileage will vary depending upon the structure and activity of your site, but any site that transmits graphics will make good use of an outbound caching content server like ISA Server.

Policy-based configuration Like all high-end firewalls, ISA Server's rules are based on abstract policies built from objects (called "Policy Elements" in ISA parlance). Objects include destinations, users, timeframes, protocols, and so forth.

There is no single place where you can take a look at the entire ruleset, however, and it's confusing to determine exactly how ISA Server is going to apply rules. For example, with protocol-based rules, ISA Server applies a "best fit" methodology with a deny-by-default policy, whereas for user groups it applies a "top-down first wins" methodology with an undeletable bottom rule that forms the default. This leads to a somewhat incoherent view of what's really going on that makes it difficult to survey an existing policy on a machine.

SSL bridging An unusual feature of ISA server is its SSL bridging capability. An ISA Server can initiate or terminate any SSL connection that flows through it. For example, if for some reason you don't want SSL flowing through the interior of your network, you can create a rule that will cause the ISA Server to respond to interior HTTP requests by creating an HTTPS connection to a secure public server. I can't figure out why this would be a good thing, but there are other purposes for SSL bridging. Another example would be connecting branch office users to an intranet at another site by creating an SSL bridge automatically between the ISA border servers. But then you'd probably just use VPN for this anyway. You can also have a front-line array of ISA servers terminate SSL in front of a secure web server to reduce its compute load. I suppose you could terminate outbound SSL connections at the firewall, inspect them, and then recreate them on the way out, so that users would have no idea that you are able to filter their secure connections. That must be what this is really for.

High availability and load balancing ISA Server is capable of using Windows 2000

Advanced Server's high-availability and load-balancing services to protect against the failure of firewalls. By configuring border ISA Servers in an array and using the built-in load-balancing facility of Windows 2000 Advance Server, you could theoretically withstand the loss of all but one of your servers and still remain online.

Quality of service bandwidth allocation ISA Server also provides a management interface for the Quality of Service scheduler (QoS) built into all versions of Windows 2000. By abstracting bandwidth into its policy schema, ISA Server makes it easy to manage the bandwidth services of Windows 2000.

Enterprise Management Features By integrating ISA Server Enterprise Edition with Active Directory, Microsoft has made it possible to publish ISA Server policies to the Active Directory for replication throughout the Enterprise. There's a bit of a chicken-and-egg problem to deal with if you intend to use an ISA Server-based VPN to connect your enterprise, but once you've got data flowing, (and assuming you have an Active Directory schema set up and running) further updates are relatively simple.

Security

The security improvements in ISA Server are dramatic when compared to Proxy Server. Proxy Server came with an anemic recommendation to "lock down the host server," without so much as bothering to explain what that entailed. Proxy Server ran as a service and literally did nothing to secure the base operating system against exploitation. Beyond that, Proxy Server was capable of proxying only HTTP and acting as a generic SOCKS proxy for other applications. While this did disconnect the direct route through the proxy and eliminate the passage of Layer-3 packets through the proxy, it did nothing to secure against higher layer attacks.

ISA Server, on the other hand, includes a simple wizard to lock down the server based on your firewalling needs, which configures the built-in security features of Windows 2000 to conform to the level of security required of a firewall. It also automatically configures the NAT and VPN features of the operating system to conform to the policies you set in the ISA Server interface, so that you have a single unified security interface on the firewall.

Basing their firewall on a complete general-purpose operating system like Windows 2000 provides Microsoft with a feature set that is second to none in the firewall market: ISA Server supports a rich set of authentication methods, installable IPsec encryption protocols, NAT, and a host of other minor features. But the foundation upon a general-purpose operating system is also the potential Achilles' heel of the firewall. Unlike other firewalls, Windows 2000 is the subject of an enormous hacking effort, and failures of its core TCP/IP networking stack are highly likely to affect ISA Server as well.

But Microsoft has done a very good job of integrating Application-layer security. ISA Server was capable of defending against the Code Red virus the day it came out, assuming your access policies were strict. And while NIMDA made it through the majority of ISA installations, it did so as an e-mail attachment that a strong e-mail stripping policy could have prevented. Few other firewalls in this book were proof against these attacks.

Finally, ISA Server supports third-party content modules and includes an SDK for developing your own in-house security filters and proxies. This provides the ability for those seriously interested in security to take matters into their own hands in a way that only open-source firewalls have allowed until now.

Interface

The user interface in ISA Server is the standard Microsoft Management Console that is used to configure all Windows 2000 administrative tools. As such, it's intuitive and easy to figure out for anyone familiar with Windows 2000 administration. The interface looks a lot like that of Symantec Enterprise Firewall, but the functionality is quite different.

The basic interface shows a tree consisting of all ISA Servers in your enterprise. Below the view of each server is located access policies, object definitions, forwarding rules, and so forth. Unfortunately, having force-fit the policy view into a tree view, there's no room for viewing all access policies and then viewing which servers have those policies applied. The interface separates the pool of ISA servers rather than joining them into a single coherent policy the way that Checkpoint Firewall-1 does.

Figure 17.3 shows the Microsoft ISA Server's MMC interface.

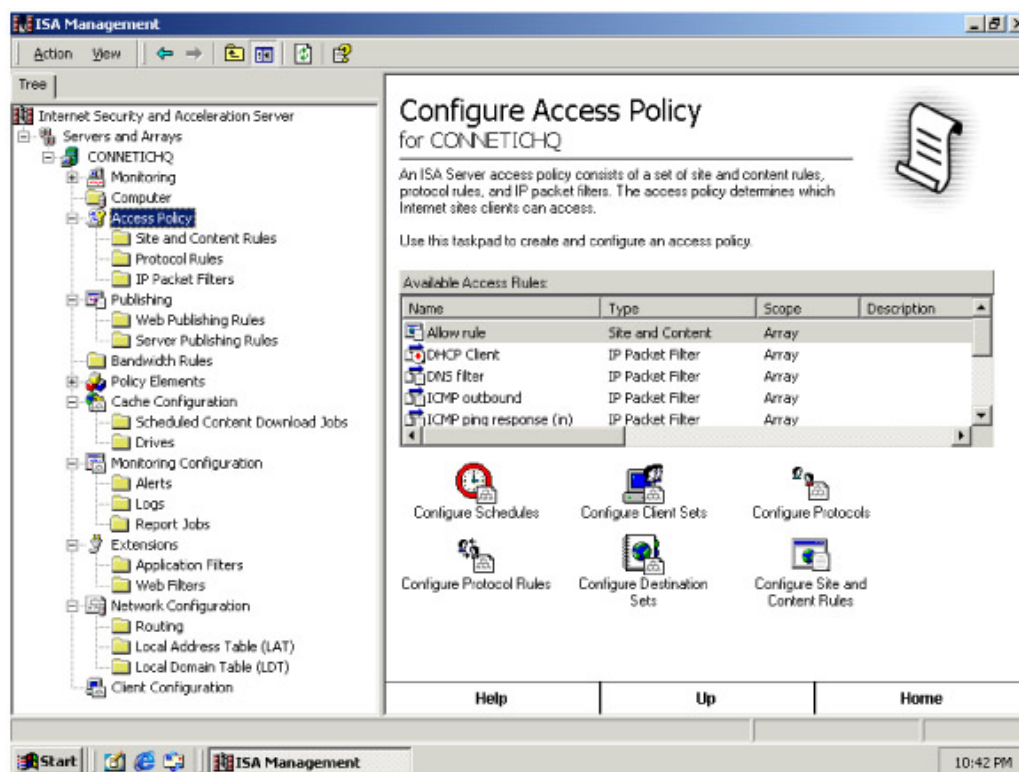


Figure 17.3: Microsoft ISA Server

Cost and Support

ISA Server is sold in two editions: Standard and Enterprise. The only difference between them is that the Standard edition is limited to running on a single machine with up to four processors, so it does not support firewall arrays, policy enforcement on remote machines, or any of the other enterprise-management features. The Enterprise edition has none of these limitations.

Like many of Microsoft's newer BackOffice products, ISA Server is licensed per processor, which means you pay a licensing fee for every processor in the machines you're using.

The Standard edition costs \$1500 per processor, while the Enterprise edition costs \$6000 per processor.

Microsoft support is the standard \$250 per call, with one free call during the first 90 days. The product is fairly easy to set up and configure, so a support call should not be necessary.

Case Study: Windows As a Firewall

Device-based firewalls are killing the traditional server-based firewall market because they're easier to buy, configure, install, and operate than their bulkier server-based products—plus, they tend to cost the same as firewall software alone, and the hardware upon which they run is usually more reliable than a PC-based server. These advantages add up to quick extinction for the server-based firewall market.

Responding to this threat, both NAI Gauntlet and Symantec Enterprise Firewall are available as devices. NAI Gauntlet can be ordered pre-installed on a Sun Netra 1U server, and Symantec Enterprise Firewall is sold as the Symantec VelociRaptor Firewall device on a Sun Raq 1U box running Linux. While convenient, these devices are more expensive than their true device-based brethren and still contain hard disk drives, which makes them more susceptible to failure.

Two years from now, I expect that the only firewalls you'll be able to buy for Windows will be the venerable Firewall-1 (before its sagging sales make it a takeover target for the voracious appetite of Computer Associates) and Microsoft's ISA Server.

But I don't expect ISA Server to be around as a separate product by 2003. Microsoft will probably roll ISA Server's features into the next version of Windows Server in order to shore up Windows' much-publicized security failures. This will allow them to publicize their server operating system as a strong firewall all by itself, and will alleviate the concern that many companies have about deploying .NET XML services on the Windows platform that is directly connected to the Internet.

Once Microsoft rolls ISA Server into the standard version of Windows, there won't be any real reason to use anyone else's Windows-based firewall on that platform. With the strong support for stateful packet inspection built into Linux now, I can think of no reason why anyone would pay for a server-based firewall a few short years from now.

That is how things always should have been.

Chapter 18: Unix Firewalls

A version of Unix exists for every microprocessor being mass-produced today and for nearly every type of computer. Unix is the closest thing to a universal operating system that has ever existed. You can load many kinds of Unix (Solaris, Linux, BSD, etc.) on your PC, you can get OS X, Linux, or BSD for your Macintosh, you can run Unix on your IBM mainframe or your Cray supercomputer, or for your VAX, if you still have a VAX. You can even get Unix for your iPAQ pocket computer.

Most commercial versions of Unix (and all the versions discussed in this chapter) are based on the original AT&T Unix, whereas most open-source Unixes are based on either the Unix derivative developed somewhat independently by the University of California at Berkeley, or on Linux, a completely independent version of the Unix operating system that was designed to be compatible with both AT&T's Unix and Berkeley's Unix. A program written for one version of Unix will probably compile and run on another version of Unix with just a little porting effort, so if you're looking for a firewall for your specific brand of high-performance workstation, you might find it in this chapter.

Computer Associates eTrust Firewall

In really big networks containing hundreds or thousands of computers, the task of administering to all those clients and servers can be overwhelming. Computer Associates developed the Unicenter TNG suite of tools to help network administrators centrally administer to a large number of network devices, including client workstations, file servers, messaging servers, network devices, routers, and firewalls. The portion that implements a firewall for Unicenter-managed networks is the eTrust firewall, formerly designated the Network Security Option for Unicenter TNG, or GuardIT.

The eTrust firewall runs on various versions of Unix and on Windows NT. Unicenter provides for centralized management of multiple eTrust firewalls distributed throughout your enterprise, providing ease of configuration and use as well as a consistent security policy for your network. Because eTrust ties into the rest of the Unicenter resource management tools, you can combine user authentication and resource access rules with the typical address and port restrictions of packet filtering.

The eTrust firewall provides stateful packet inspection, Network Address Translation, packet inspection and rewriting for supported protocols, generic proxying for redirectable protocols, and centralized authentication. The sophisticated security event monitoring, logging, and response features of this firewall even allow for automatic reconfiguration of the security policy when suspicious or threatening activity is detected, which allows the system to lock itself down and gives you time to respond to the problem.

Pros	Cons
Runs on Unix and NT	Cost
Integrates with Unicenter	Requires Unicenter TNG
Centralized management	Long Learning Curve
Strong remote management	
Fast and flexible	

The platform requirements are as follows:

- Intel Pentium Microprocessor or Unix workstation of equivalent power

- 64MB RAM (128MB recommended)
- 500MB hard disk drive, additional for caching
- Unix or NT
- At least two network interfaces

Major Feature Set

The major features of eTrust include the following:

- Packet Filter (stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Proxies (HTTP, FTP, RealAudio, etc.)
- Transparent proxies
- Reverse proxies (HTTP, SMTP, FTP, etc.)
- Secure authentication (NT Server, RADIUS Server)
- Logging to databases and e-mail notification

The included stateful inspection filter is very strong and comparable to the stateful inspection services provided by Checkpoint Firewall-1. Network Address Translation is built into the stateful inspector.

The proxy functionality of eTrust doesn't really occur at the Application layer; protocol payloads are rewritten directly by the stateful inspector rather than being handed off to a separate Application layer service, which regenerates the connection in its entirety. Rewriting provides much the same benefit; portions of the protocol that the firewall doesn't know about can't be rewritten, and such parameters as proper buffer length can be checked to prevent buffer overrun conditions.

Minor Feature Set

Some of the minor features of eTrust include the following:

- Content filtering (Java, Virus Scanning, URL blocking) through the additional eTrust Content Inspection and eTrust AntiVirus packages
- Scan detection, spoofing detection, and automatic blocking through the additional eTrust Intrusion Detection package
- Graphical administration
- Remote administration
- Centralized administration
- Integration with overall enterprise management tools
- Transparent ARP support
- SYN flood protection
- Anti-spoofing control
- Real-time monitoring and reporting
- Policy-based configuration and management
- Calendar support

A central policy-based management application (Unicenter TNG) provides strong centralized management for the firewall. Policies can easily be created and applied across the enterprise from the Unicenter control application. Unicenter TNG also provides a platform for strong integration with the other IT management options available for the system and provides the foundation for the log,

alert, event detection, and response features.

The calendar support of the eTrust firewall is a useful feature that allows you to change the firewall policy based on the time. For example, you could significantly restrict outbound communications from your protected LAN after working hours, when users are not expected to be using the network. Violations can be logged and investigated as potentially compromised computers opening a back channel to outside hosts.

Interface

With eTrust, there is a graphical interface for both Windows NT and for Unix. Firewalls appear as resources to be administered from the Unicenter administration suite. Because the eTrust uses the same framework as all of the other Unicenter options, administrators in a Unicenter shop will find the interface to be friendly and comfortable.

The graphical interface makes it easy to set up rules and enable or disable specific services for particular computers or users. The security objects are integrated with the other components of the Unicenter system (such as the Single Log On option), sparing you the effort of both establishing user account information and recording security restrictions in multiple locations.

Security

The eTrust firewall uses a stateful inspection packet filter, which keeps track of connection information across multiple packets. These include UDP packets, which do not retain session information. The packet filter checks all the typical IP packet features such as source and destination addresses, port numbers, options set, SYN bit, ICMP messages, and so on. In addition, the packet filter can integrate into its rule set additional information obtained from the rest of the Unicenter framework, including user identity, allowed access times, and network resource restrictions. The firewall checks every packet before the IP stack processes it, thereby blocking attacks against the firewall itself using malformed and maliciously constructed IP packets, such as the Ping of Death, teardrop attacks, and so on.

One performance advantage of the firewall is that it can perform the equivalent of protocol proxying for some protocols by directly manipulating the IP packets, rather than handing the packets off to a separate proxy server application. This provides for much faster proxying and therefore increased throughput and reduced latency between your network and the Internet. The firewall also provides for generic port redirection and integration with the Internet Web Management option to Unicenter TNG.

Documentation, Cost, and Support

Using eTrust requires a Unicenter TNG network infrastructure, which is designed for larger businesses. Because pricing varies widely and depends largely upon your Unicenter infrastructure, there's no meaningful way for us to provide pricing information. Contact a CA sales representative directly to obtain pricing information if you use or want to use Unicenter TNG.

Tip You can get more information about Unicenter TNG at <http://www.cai.com/>.

SecurIT Firewall

The SecurIT firewall from SLM (formerly MilkyWay) is available for both Unix and NT. This firewall, like the free TIS FWTK described in Chapter 16, does not perform any packet filtering. Instead it provides Application-level proxies for each of the protocols that will pass from the internal network to the Internet. Also like FWTK, the SecurIT firewall uses authentication to provide user-based as well as IP address-based access control. Where SecurIT really shines, however, is in the wide variety of protocols it "scrubs" or provides proxy redirection for. In addition to the proxies, SecurIT has a strong VPN component that allows you to establish encrypted IP tunnels between your protected LANs over the Internet.

•	
Pros	Cons
Runs on Unix and NT	No packet filtering
Supports a wide range of protocols	NT Version does not harden OS
VPN	Cost
Centralized authentication	Difficult to acquire
High speed application proxying	

Platform requirements for SecurIT Firewall include the following:

- SunSparc 5 or any Ultra-SPARC, Intel Pentium
- 2GB hard disk drive
- 32MB RAM
- PCI Quad adapter
- 2 or more network cards
- CD-ROM drive

Major Feature Set

SecurIT provides the following major features:

- Bidirectional transparent proxy services for a wide variety of protocols
- VPN between SecurIT protected networks
- DMZ Support
- Secure authentication (Unix passwords, S/Key software, SecureID, Safeword Enigma Logic)
- Logging to databases and e-mail notification

SecurIT provides numerous security proxies for common Internet protocols, which makes its protocol security very strong.

SecurIT uses its generic TCP proxy functionality to perform client hiding, a function their documentation calls Network Address Translation. The functionality is not equivalent to true Network-layer NAT.

Secure authentication is performed via Bellcore's (now Telcordia's) S/Key one-time-password algorithm.

Conspicuously missing from the major feature set are packet filtering and Network Address

Translation. Neither function is necessary in a strong security proxy as long as the base operating system is sufficiently hardened. Neither Solaris nor NT is hardened in our opinion, and this considerably weakens the ability of firewalls that do not implement their own packet filtering accordingly. SecurIT ships with a version of Solaris that has apparently been hardened, and recommends security patches as additional vulnerabilities are discovered, but the NT version is susceptible to a wide range of denial-of-service attacks.

Minor Feature Set

SecurIT provides the following minor features:

- SQL proxying
- Remote administration
- Content filtering (Java, Virus scanning, URL blocking, etc.)

As with most true firewalls, SecurIT is capable of logging to databases and transmitting e-mail to alert on security events. A SQL security proxy is provided to support SQL*Net transactions through the firewall.

Security

SecurIT does not filter packets before they are delivered to the IP stack for processing. The firewall relies on the underlying operating system to be resistant to IP-level attacks. Both Solaris and Windows, at their most current patch or service pack, have finally been made highly resistant to known attacks, but undiscovered vulnerabilities almost certainly exist in both operating systems. SecurIT for Solaris ships with a hardened version of Solaris.

Instead, SecurIT is a proxy server, which examines the data portions of IP packets to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example).

SecurIT is designed with performance in mind. This highly optimized proxy server uses threads and shared memory to minimize the time required to filter the proxied protocols, allowing more traffic to pass through the firewall, while still fully examining all of the data to ensure that it conforms to protocol specifications.

SecurIT comes with a number of application-specific firewall proxies. In addition to providing content filtering for the specific protocol (guaranteeing that the port is actually used by the appropriate protocol instead of some other program), each protocol can be configured to block certain IP addresses and Internet domains. SecurIT provides proxies for the following protocols:

- FTP—A standard FTP service proxy.
- Generic SOCKS—Allows the administrator to redirect easily proxied protocols by specifying the address and port to forward TCP and UDP packets to.
- Gopher—Proxies the text-based hypertext protocol that (barely) predates the Web.
- HTTP++—Allows basic web traffic, but allows the administrator to block applets and URLs.
- HTTP—For basic port 80 proxying or for web traffic on other ports, but using the HTTP protocol.
- LDAP—Allows network clients to access directory servers exterior to your firewall.
- Mail—Stores and forwards e-mail delivered to the firewall for delivery on your local network.
- NNTP—Forwards Usenet news through the firewall.
- POP—Provides a channel for internal clients to access external e-mail servers.

- Real Media—Channels audio and video conforming to the Real Media standard through the firewall.
- RPC—Provides for secure Remote Procedure Call through the firewall.
- SSL—Forwards secure socket communication through the firewall.
- Telnet—Proxies command-line control of remote computers.
- VDO Live—Mediates VDO multimedia from internal clients to external multimedia servers.

Documentation, Cost, and Support

The SecurIT firewall is sold by the number of open simultaneous connections (sessions) rather than the number of IP addresses inside the network. This means, for example, that a 15-user network could probably get away with a 10-session version of the firewall if only 66 percent of the users were using the Internet at any one time. Prices shown are for the Solaris edition with one year of included support. The U.S. distributor would not quote pricing for the Windows version, as they considered the Windows operating system to be nonsecure. The product is sold primarily to military and government channels since SLM has no significant marketing through commercial channels.

The product ships with a hardened version of Solaris so there's no need to purchase the operating system. Hardware costs for a Sun Ultra-5 run about \$5,000.

- 10 sessions: \$3,600
- 40 sessions: \$7,200
- 100 sessions: \$16,200
- Unlimited: \$23,400
- VPN: +\$1,200

Tip You can browse SLM's website at slmsoft.com. To purchase SecurIT, contact Neoteric at (212) 625-9300.

NetWall

Group Bull, a major European manufacturer of electronics and software has packaged their internal IP security expertise into a firewall product called NetWall. This firewall runs on Sun's Solaris and IBM's AIX versions of Unix as well as Windows. The secure remote control software for the firewall runs on Windows platforms as well as AIX.

NetWall gives you the full range of security options to work with—from stateful packet inspection to Application-level proxies for a wide variety of protocols, NAT, VPN, authentication, load balancing, remote control, and support for third-party content inspectors thrown in as well.

•	
Pros	Cons
High speed	Cost
High reliability	
Centralized authentication	
Versatile proxying	

NetWall suffers from a difficult setup and a lack of integration among software components. Configuring the firewall is not particularly easy compared to the majority of firewall offerings in this book.

Major Feature Set

NetWall offers the following major features:

- Packet filter (Stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Transparent and reverse proxies (SOCKS (IP, IPX), HTTP, Telnet, Gopher, SMTP, FTP, POP, IMAP, RealAudio/Video, H.323)
- Secure authentication (Plain, Proprietary, Security Dynamics, Crypto Card, S/Key, RACAL, RADIUS Server)
- VPN (Proprietary)
- VPN client software (Windows 98/NT/2000/XP)
- Redirection for load balancing and high availability through add-in package
- Firewall high availability
- Logging to databases and e-mail notification

NetWall includes a strong stateful inspection filter and Network Address Translator that supports both static and dynamic address mapping.

NetWall supports a broad range of authentication features, including low security options like ASCII plain text passwords and higher security options like RADIUS, MD/5 Challenge/Response, Bellcore S/Key one-time passwords, SecurID Cards, and smart cards. NetWall also includes a complete set of APIs to allow third-party vendors or organizations with programming support to create other authentication options.

The remote access VPN is different than the firewall-to-firewall VPN. The remote access VPN is somewhat unique in that it is based on a SOCKS proxy transmitted through an SSL tunnel, rather than IPsec. The remote access VPN supports standard 40-, 56-, and 128-bit key lengths. The firewall-to-firewall VPN is based on DES and triple-DES, and supports key lengths up to 192 bits.

Minor Feature Set

NetWall offers the following minor features:

- Content filtering (Java, Virus scanning, URL blocking) through MimeSweeper and VirusWall plug-ins
- Graphical administration
- Remote administration
- Centralized administration
- OS hardening
- SQL proxying

Multiple NetWall firewalls can be used to balance the connection load between them and to continue operating in the event that one of them fails. This allows you to provide high availability of Internet services and protects you in the event of a denial-of-service attack.

NetWall supports content vectoring to third-party content scanning applications such as MimeSweeper or VirusWall.

Firewall management can be performed remotely from any Windows or AIX workstation. Communications between the firewall and the management workstation is encrypted.

Interface

NetWall's GUI interface is typical of policy-based firewall managers, providing a similar look and feel as Checkpoint Firewall-1's interface. As with Firewall-1, the interface can be run locally on the firewall or on a remote management workstation.

Security

NetWall's IP filter performs stateful packet inspection, keeping track of the state of TCP and UDP data streams. (The state mechanism allows the firewall to keep track of UDP in spite of the fact that UDP doesn't keep session information in the packets.) The NetWall packet-inspection engine can also inspect the data portion of some IP packets directly, which simplifies and improves the proxying performance of certain protocols. Protocol filters that the IP filter accelerates include HTTP, SMTP, FTP, Telnet, RPC, SQL* Net, and SAP.

While the IP filter accelerates the proxies and protects the firewall server from IP-level attacks, the application proxies make sure that only safe data traffic transits your firewall. NetWall comes with an impressive range of proxies, including the following:

- FTP—Filters FTP traffic.
- Generic—Allows the administrator to redirect easily proxied protocols by specifying the address and port to which TCP and UDP packets should be forwarded.
- Gopher—Proxies the non-multimedia hypertext protocol that (barely) predates the Web.
- HTTP—Proxies for basic port 80 or for web traffic on other ports, but using the HTTP protocol.
- SHTTP/SSL—Proxies for encrypted web traffic and for Secure Socket Layer communication.
- LDAP—Allows network clients to access directory servers exterior to your firewall.
- SMTP—Stores and forwards e-mail delivered to the firewall for delivery on your local network.
- IMAP4—Mediates mail delivery and mailbox checking through the firewall.
- NNTP—Forwards Usenet news through the firewall.
- POP3—Provides a channel for internal clients to access external e-mail servers.
- Real Audio/Video—Channels audio and video conforming to the Real Media standard through the firewall (AIX version only).
- H.323—Allows for videoconferencing through the firewall (AIX version only).
- Telnet—Proxies command-line control of remote computers.
- TN3270—Proxies TCP/IP access to mainframe and minicomputers.
- TNVIP—Allows TNVIP access across the firewall.
- SOCKSV5—Redirects protocols specifically designed to be redirected through the SOCKS proxy service.

Documentation, Cost, and Support

Bull has firmed up distribution of their firewall in the U.S. since the first edition of this book. Contact www.evidian.com/accessmaster/netwall for more information and contact information regarding this product.

Network Associates Gauntlet on the WebShield e-ppliance

Network Associates (NAI), the new owners of Gauntlet, is the result of the merge between McAfee (of virus-scanning fame) and Network General (makers of the Sniffer network protocol analyzer). The company then purchased PGP, Phil Zimmerman's encryption technology company, and Trusted Information Systems (TIS), the makers of Gauntlet. TIS developed the first security proxies under contract to the Department of Defense's Advanced Research Projects Agency (DARPA) when DARPA decided that stateless packet filters were not effective security devices. These original TIS security proxies are still available at no charge on the Internet and were covered in Chapter 16.

NAI has put Gauntlet in the PGP group and is in the process of merging all their security products together through a mechanism that they call "Active Security." Active security is an event-driven publish/subscribe mechanism that allows the various software components of a security infrastructure to report exceptional events to other components in the security group. The security components are then able to make adjustments to their security policy to deal with the changed circumstance.

The level of conformance to this new active security infrastructure is low—most products can do little more than report events. But it does show that NAI is serious about integrating their security products, and that they understand how it needs to be done. No other security product vendor has shown as much understanding of total security than NAI in this respect. How much of this talk becomes reality, and how useful the product will be when it does, remains to be seen.

Gauntlet is widely regarded in the security industry to be the most secure firewall on the market, because it uses security proxies for all secured services rather than relying on stateful packet inspection. Recent versions include support for adaptive filtering, whereby connections are inspected at the Application layer by a proxy server during initiation, and then dropped down to the Network layer for stateful filtering once the connection is established and authenticated. This improves the performance of the firewall dramatically.

Gauntlet is available for Windows NT and Unix. The firewall is multithreaded, which means it provides higher performance on multiprocessor machines. In addition to being sold as a software package, you can purchase software that has been pre-installed on a Sun workstation and sold as an "e-ppliance." This turns the Gauntlet software into a firewall device much like those examined in the next chapter, however it contains hard drives (therefore it is not solid-state and is more likely to fail eventually) and it contains a feature-rich operating environment that hackers can use in the unlikely event that they penetrate the Gauntlet firewall security.

-

Pros

- High speed
- High reliability
- VPN Support
- Centralized authentication
- Cross-platform & device packaged

Cons

- High cost

Major Feature Set

- Gauntlet provides the basic components required of a modern firewall: Packet Filter

- (stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Transparent and reverse Proxies
- Secure authentication (SecureID, RADIUS S/KEY, CryptoCard, ActiveCard, Microsoft Windows NT Challenge/Response)
- VPN (IPSec/IKE accelerated)
- VPN Client Software (Windows 98/NT/2000/XP, Macintosh, Unix, Linux)
- Redirection for load balancing and high availability
- Firewall High Availability
- Bandwidth control and Quality of Service
- Logging to databases and e-mail notification

The Gauntlet Packet Filter

Gauntlet is now a combination of a security proxy and a stateful inspection filter. Each time a connection is established, the initial connection establishment packets are transmitted through the application proxy. Depending upon the security settings established by the security administrator, the proxy can continue to proxy all the data in the connection or determine that the connection is trustable and direct the packet filter to simply forward remaining packets in the connection without further inspection through the proxy. This approach lessens the rather serious performance and load problems from which security proxies suffer, but retains most of the security provided by an application proxy.

Proxy Services

Gauntlet provides support for an impressive range of both traditional Internet services and the newer multimedia and database services.

Standard Internet services include:

- FTP
- HTTP
- LDAP
- NNTP
- POP3
- PPTP
- SMTP
- SNMP
- SSL
- Telnet

H.323 Multimedia services include:

- NetMeeting
- NetShow
- RealAudio
- RealVideo
- VDOLive

SQL services include:

- Microsoft
- Oracle
- Sybase

Network Address Translation (NAT)

Illegal Network Address Translation is NAT in an environment where valid IP addresses (those not in the 10, 192.168, or 176 domain) assigned to other owners are in use in your network and must be translated to legal addresses for proper operation on the Internet. Illegal network address translators can deal with the special problems posed by illegal addresses.

Tip Reconfigure your entire network to use the legal, non-routable 10 domain for internal addresses. There are illegal address problems that INAT translators can't solve. I've had customers who have tried for years to deal with their illegal address schemes rather than put in the few days of intense, IT organization-wide effort it would take to rebuild the address infrastructure using DHCP and legal addresses. If your network is so encrusted that you don't dare change IP addresses, it's a disaster waiting to happen anyway.

Minor Feature Set

Gauntlet supports the following minor features:

- Content filtering (Java, virus scanning, URL blocking)
- Scan detection, spoofing detection, and automatic blocking
- Graphical administration
- Remote administration
- Centralized administration
- Transparent ARP support
- SYN flood protection
- Anti-spoofing control
- Real-time monitoring and reporting
- Content Vectoring Protocol
- Policy-based configuration and management
- High performance
- OS hardening
- SQL proxying

The Gauntlet Content Vectoring Protocol allows firewall vendors and third-party providers to create connectable content scanners through which certain types of content must pass before they clear the firewall. Gauntlet's virus protection and Java filtering are performed using CVP technology.

NAI uses the CVP to tie the Gauntlet firewall in to their own McAfee content-filtering service, which periodically updates the firewall with new fingerprint files for viruses, Trojan horses, and other malicious content that may attempt to transit your firewall in HTTP, SMTP, FTP, or some other protocol. Gauntlet is an attractive solution for networks that already use the very popular McAfee virus-scanning software on workstations inside their network.

Security

Gauntlet secures the firewall with various policies that are actually accumulations of service proxy rules. You are free to develop as many different policies as you need. These policies are then mapped to network interfaces for implementation. Two default policies are created when the product

is installed:

- Trusted policies are mapped to network adapters that you identified as "inside" during the installation process.
- Untrusted policies are mapped to interfaces you identified as "outside" during the installation process.

The trusted policy engages the following proxy services:

- FTP
- H.323
- HTTP
- LDAP
- NetShow
- NNTP
- PPTP
- RAP
- SMTP
- StreamWorks
- Telnet
- VDOLive

The default trusted policy disallows the following proxy services:

- MS-SQL
- POP3
- SNMP
- SQL-GW
- Sybase-SQL

The effect of this policy is to allow most normal consumption content to pass through the firewall, but to block attempts by internal clients to interact directly with foreign untrusted e-mail servers or SQL servers.

The default untrusted policy allows the following services:

- FTP
- NNTP
- POP3
- SMTP
- Telnet

The policy disallows all others. This allows untrusted hosts to make FTP, NNTP, e-mail, and Telnet connections inside the network. Authentication is required for all of these services.

Interface

The interface is clean and simple—much less cartoonish than either Guardian or Firewall-1. The firewall manager application uses a tabbed view to switch between the various dialogs used to configure the firewall.

Documentation

All documentation included with the evaluation edition of Gauntlet comes in the form of the help file for the firewall manager. This makes the documentation somewhat difficult to read straight through, but easier to read in a digressional click-through manner. HTML-based documentation would have combined the best of both worlds.

The documentation is light, focusing on simple explanations of broad security concepts and relying upon the administrator to figure out the technical nuances of firewall operation.

Cost and Support

Cost information about NAI Gauntlet can only be obtained on a per-configuration basis from a Gauntlet sales representative, which makes it very difficult to compare the firewall's price against the competition.

The device we tested for this book, an unlimited e-appliance with VPN, weighed in at a hefty \$17,500, making it by far the most expensive firewall we tested. Only ISP-grade, high-speed firewalls cost more.

Evaluation editions of Gauntlet firewall can be downloaded at <http://www.nai.com/>.

SunScreen Secure Net 3.1

All the big information technology companies have crafted their own firewall software, and Sun is no exception. They have a firewall called (cleverly enough) SunScreen, which runs on Sun SPARC workstations and Intel computers under Solaris and is expertly designed for providing high-throughput protection for large networks.

-

Pros	Cons
High speed	High cost
High reliability	No Content filtering
VPN support	
Centralized authentication	
Java-based administration	

Note In addition to selling their own firewall software, Sun provides the hardware for the appliance versions of Gauntlet, Raptor, and SLM firewalls.

Major Feature Set

SunScreen provides the following major features:

- Packet Filter (stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Proxies (HTTP, SMTP, FTP, Telnet)
- Secure authentication (Plain, SecureID, RADIUS, SKIP)

- VPN (IPSec/IKE, SKIP)
- VPN client software (Windows 98/NT/2000/XP, Macintosh, Unix, Linux)
- Firewall high availability
- Bandwidth control and quality of service
- Logging and e-mail notification

Sun's firewall provides all the major features you'd expect in a network firewall, which is to be expected from the company that coined the phrase "The network is the computer." Also gratifying is the use of industry-standard protocols for VPN support, so you can connect any client for whom you have IPSec software to your LAN. The firewall doesn't give you anything you can't get in other firewalls though, so the best reason to install SunScreen is if you're already a Sun shop.

Minor Feature Set

SunScreen provides the following minor features:

- Scan detection, spoofing detection, and automatic blocking
- Graphical administration
- Remote administration
- Centralized administration
- Transparent ARP support
- SYN flood protection
- Anti-spoofing control
- Policy-based configuration and management
- High performance
- OS hardening
- Stealth

The interesting feature here is the SunScreen support for "stealth" or "drop in" fire-walling, where the firewall interposes itself between the hosts on a network and the outside network. In stealth mode, the firewall does not present any IP addresses to either the LAN or to the WAN connection, instead it monitors all network traffic on its interfaces and it forwards packets or does not forward them as appropriate. The firewall itself is not addressable and therefore is unsusceptible to most firewall subversion methods. See the WatchGuard Firebox 1000 in Chapter 19 for another firewall that can operate in this manner.

One noticeable lack in the SunScreen software is built-in support for content filtering. The SunScreen proxies ensure that you're actually using HTTP on the HTTP port, but they do not keep you from downloading a virus. Perhaps this is a reflection of Sun's Unix culture, which is far less susceptible to the viruses, malicious ActiveX controls, and Visual Basic scripting worms that plague Windows networks. Nevertheless, if you need to protect Windows computers, you should consider adding a virus scanner to your network in addition to the SunScreen firewall, and there are many virus scanner products that will run on Solaris alongside the firewall.

Interface

Centralized management of multiple bastion hosts is performed via a Java applet running in web browsers. This makes administration very flexible, because you can administer the firewall from any Java-enabled browser with SKIP installed. It's no surprise that Sun would provide this option because Sun developed Java and is Java-enabling all of their enterprise software.

The interface is clean, simple, and makes good use of Java technology. Management is policy based. Figure 18.1 shows the initial policy page.

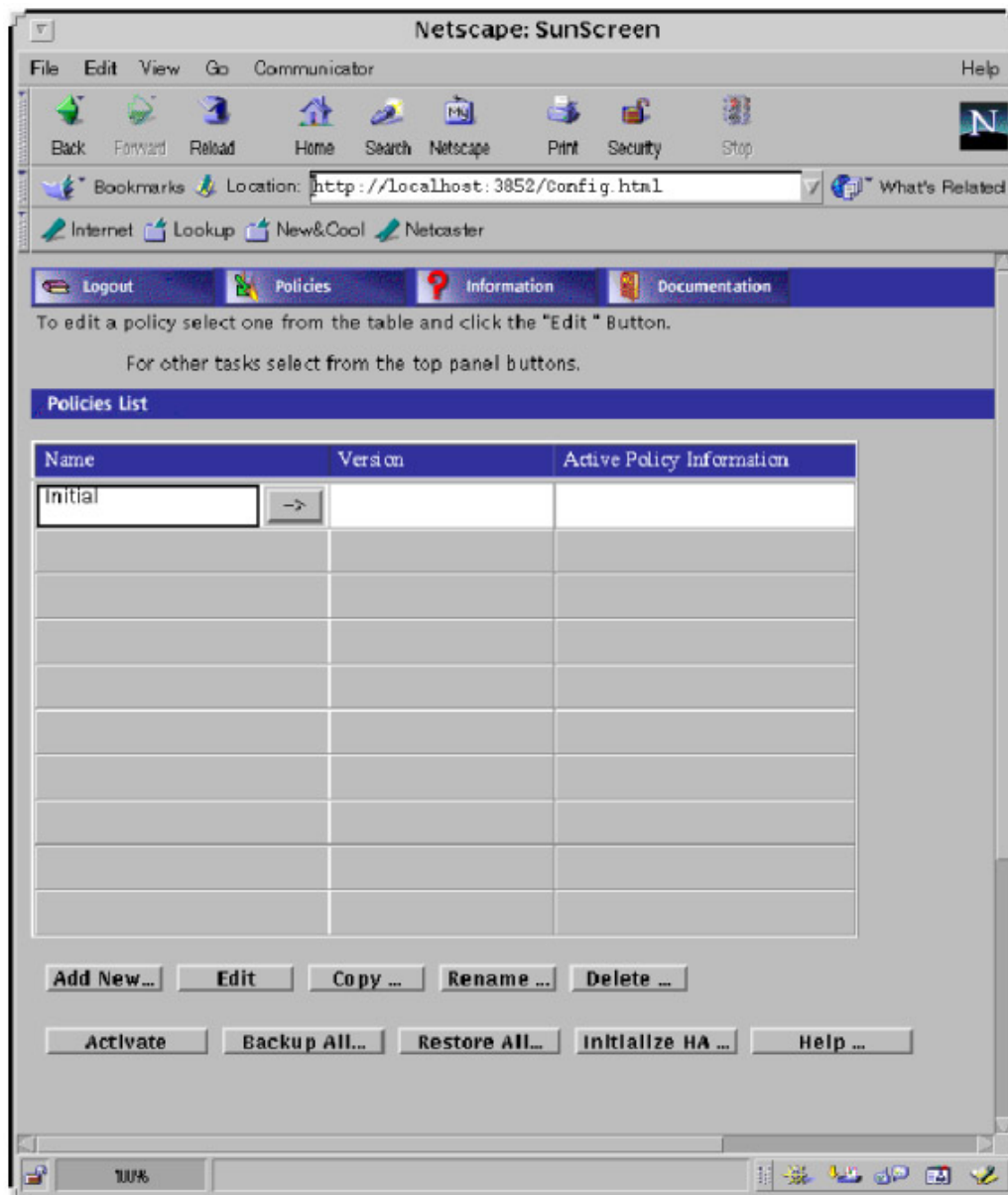


Figure 18.1: The SunScreen Initial Policy page

Security

SunScreen hardens a Sun SPARC or Intel workstation running Solaris to perform as a packet filter and Network Address Translator. All packets are processed by the packet filter before being routed or translated. SunScreen provides the full range of packet filtering options, including the SYN bit, source and destination IP addresses, source and destination ports, packet type, and so on.

Because SunScreen does not inspect the data portions of the packets and locks down the operating system (so naïve administrators can't compromise security by running insecure services on the server), you will need a proxy server running on another computer to ensure that the traffic traversing a particular port conforms to the protocol for that port (that only HTTP requests and replies are going over port 80, for example). Many web servers will also act as HTTP proxies, and you can use servers for store-and-forward protocols (such as SMTP and NNTP) unmodified as protocol proxies for their services. Ideally, you should use address translation to redirect the

appropriate traffic to and from these servers.

SunScreen evaluates every packet received by the network adapters in the firewall computer according to a set of rules you establish from the Java administration console. The rules are applied in order and one at a time until SunScreen finds a rule that matches the packet and specifies a terminal action, such as ACCEPT or DROP. Because the rules are applied in order, it is vitally important to craft the rules in the right order.

Documentation, Cost, and Support

SunScreen EFS is sold on a per-user basis, with VPN licensing as a separate cost.

- SunScreen EFS 3.0, unlimited users single server: \$10,000
- 250 Client licenses for SunScreen SKIP: \$10,000
- SSN 3.0 Competitive Upgrade: \$3,000
- SSN 3.0 for Workgroups unlimited + 100 Clients for SKIP: \$7,000
- SSN 3.0 Evaluation Kit: \$100
- SSN 3.0/WG with 100 uses +100 SKIP: \$3,000
- SSN 3.0 Site +250 SKIP Clients: \$35,000
- SSN 3.0/WG Unlimited use +250 SKIP Clients: \$70,000
- SKIP client for Windows 9x/NT (1 server, 1 user): \$150
- SKIP clients, 1000 pack: \$41,000

Tip Visit Sun's website at www.sun.com/security.

Case Study: Try to Buy

To provide cost and support information for the various firewalls in this book, I went through the same sales channels that any knowledgeable consultant would use. Primarily based on websites, I searched for sales channels for the product, contacted the contacts listed in vendors sites, and basically did whatever the company's website told me to do to acquire the firewall. I felt this approach would closely approximate the typical firewall buying experience.

Surprisingly, my survey yielded mixed results. Some firewalls were incredibly easy to buy—their websites went right through to an online store willing to take your credit card number and ship you the product the next day. Others went the more traditional route of listing numerous distributors.

I also had great success finding firewalls available from online distributors at <http://www.shopper.com/>, for those firewalls in a traditional distribution channel.

Other firewalls were so difficult to obtain pricing information for that I would have given up had I not been doing research for a book. The companies that sell these firewalls have chosen to work exclusively through value-added reseller agreements, which leads customers down a Byzantine maze of voicemail in an attempt to find product sales information.

For one product, my phone calls to the numbers listed on their web pages yielded numerous incorrect and out-of-date phone numbers. Calling their tech support reached a voicemail box, and leaving a message did not generate a return call. When I called the main number and asked for pricing information about the SecurIT Firewall, I was transferred six times until I reached a voicemail box. I received a call back from a sales representative who directed me to their primary U.S. distributor, a company that appeared to be a very small operation—they had only one sales person who was qualified to provide pricing information about the firewall.

Another product from a major multinational vendor was simply impossible to obtain U.S. pricing information for. When I contacted the company that their website had listed as their U.S. distributor, that company had no idea that they carried the product. I was then transferred to another distributor who had no idea what I was talking about. I finally just gave up.

Firewall vendors who can't figure out how to sell their product are likely to be completely unable to support it. Although I hate to make recommendations based on nontechnical criteria like sales and marketing, especially when the two firewalls that suffered from these problems are very strong security proxies, I just don't think it's worth the potential support problems you'll have with a completely non-responsive company.

Chapter 19: Device and Specialty Firewalls

Overview

There are two kinds of firewalls—software based and hardware based. The previous chapters have examined firewalls that run as applications on conventional operating systems such as Windows NT or versions of Unix. This chapter describes those firewalls that provide their own underlying operating system. With these firewalls you just turn them on, or (at most) insert a floppy disk and turn them on. Also in this chapter, we talk about a couple of firewalls that run on unusual platforms (for firewalls) such as AS/400 or NetWare.

The nicest thing about a device-based firewall is that you only worry about keeping one piece of software current—that of the firewall itself, usually in the form of a firmware update. You don't have to download operating system patches, new kernels, service packs, or security updates. This makes keeping the firewalls current considerably easier. It also gives you one vendor to point your finger at when a weakness is found.

Device-based firewalls are also often much easier to set up and get running than software-based firewalls. They arrive with the software already installed in the device, and all you have to do is give it valid IP addresses to use. Policy configuration is usually just a matter of installing and using a Windows application or web interface to manage the machine.

This chapter also covers those firewalls that run on standard computers (all PCs, actually) but do not use a standard Unix distribution or Windows NT as their host operating system. Despite the hype, Windows NT and Unix are not the only operating systems in existence. Firewalls for other operating systems abound and are, in many cases, more secure.

Because these firewalls are based on unusual operating systems, hackers have not yet created a trove of the various attacks against them, such as exploiting buffer overruns in the Unix sendmail daemon or exploiting bugs in Internet Information Server on Windows NT platforms. Many of these operating systems were uniquely developed by their vendors to support a specific firewall product, so they are completely proprietary. This lends a strong measure of "security through obscurity," and keeps the hordes of typical hackers (those who merely read and repeat known attacks rather than developing new ones) completely at bay.

Obscurity has its price, however. Almost all of this type of firewall require unique adapter drivers and will only work with specific adapter models. Patches for these firewalls are rare, so if an exploit for one of them is developed, it usually takes until the next revision of the software before it's fixed. Some of these firewalls operate on platforms with arcane user interfaces that you may not be familiar with.

These firewalls also suffer from a lack of complete features. They are either based on generic SOCKS proxies or stateful inspection, and usually do not provide any support for the opposite type of firewall. The firewalls also suffer from a generational lag behind the firewalls developed for Unix and NT because software is much harder to develop for smaller-market operating systems.

NetWare is well entrenched in the server market, and thousands of "red" (Novell-only) networks exist. Managers in these environments rightly balk at the requirement to become an expert in a foreign operating system for the sole purpose of establishing a firewall. Novell markets a very strong firewall that runs on NetWare called BorderWare for these environments.

The mainframes of yesteryear have been converted to the application servers of today. VAX and AS/400 machines running VMS and OS-400 now serve as web servers, e-mail hosts, and e-commerce engines. They also require protection, so there are firewalls available for them.

I've rolled these smaller-market operating systems together into a chapter because of the limited fields they represent. In many cases, the firewalls I profile here are the only serious firewalls available for the platform shown.

Keep in mind that your choice of application or file server doesn't constrain your choice of firewall—you can use an NT firewall in a Novell network and a Unix firewall to protect an AS/400. Because of the high cost of small-market software, it's usually more economical to use a larger-market platform for generic services like firewalling. To run an OS-400 firewall on the AS/400 will cost you tens of thousands of dollars, compared to the few thousand for a robust PC. These costs should be balanced against the cost of training administrators on an unfamiliar operating system and the security risk of operating a firewall in an environment that may not be completely familiar.

SonicWALL

If you want the no-holds-barred easiest to use firewall you can buy, get a SonicWALL. You just drop it in, point a web browser at it to configure it, and then use it. There's not a whole lot to configure, just the interface addresses and what ports you want to let in and out. If you want a VPN, you set up the shared secret IKE keys and the hosts to allow, and then, again, you just use it.

•	
Pros	Cons
No hardware or software required	No true Application-level proxying
Strong stateful inspection	
Simple configuration	
Highly reliable	
Highly compatible VPN	

SonicWALL devices are the closest things you'll find to a true plug-and-play, install, and forget firewall. For environments without on-site support staff, they are the way to go since they're very easy to manage remotely and unlikely to suffer from failures that can't be corrected remotely. We routinely update the firmware on these devices remotely and have never run into any significant problems.

Major Feature Set

The SonicWALL Firewall provides the following major features:

- Packet filter (stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Secure authentication (IPSec/IKE, certificates, RADIUS Server)
- VPN (IPSec/IKE)
- VPN Client Software (Windows 98/NT/2000/XP)

- Firewall high availability
- Logging including syslog and e-mail notification

The most obvious feature missing in the major feature set of the SonicWALL is proxy services. If you need to strip viruses from mail attachments, then you'll have to install a separate proxy server to do it.

The DMZ support includes a nice feature—the DMZ hosts supported can be configured to be in the same (public) IP subnet that the firewall itself resides in. The SonicWALL must of course be installed between the DMZ Ethernet and the public Internet connection, but that way it can transparently redirect and filter traffic between the DMZ and the Internet. With a SonicWALL, you do not set the IP address of the DMZ interface because it is set to be the same as the public interface, even though it is a physically separate connection.

Minor Feature Set

The SonicWALL Firewall supports the following minor features:

- Scan detection, spoofing detection, and automatic blocking
- Limited HTTP content filtering
- DHCP
- Graphical administration
- Remote administration
- SYN-flood protection
- Anti-spoofing control
- High performance

The nicest thing about the SonicWALL is its web interface. You don't have to install any special software to configure it, and you can manage it from any machine in your LAN that has a Java-capable web browser, including Unix or the Macintosh (which is an important feature for those few institutional holdouts that haven't caved to the Microsoft monopoly). Most other device-based firewalls require you to install Windows-specific software to control them. You can even manage the SonicWALL from outside your network if you have configured the VPN properly and enabled the feature.

Installation, Interface, and Documentation

The SonicWALL is pretty much plug-and-play, with minimal web configuration. Chapter 11, "Configuring a Real Firewall," covers SonicWALL in detail because it is the "real firewall" used in the chapter. In summary, the installation is easy, the interface is simple, and the documentation is straightforward, if a little shallow. Figure 19.1 shows the Sonic-WALL web configuration interface.

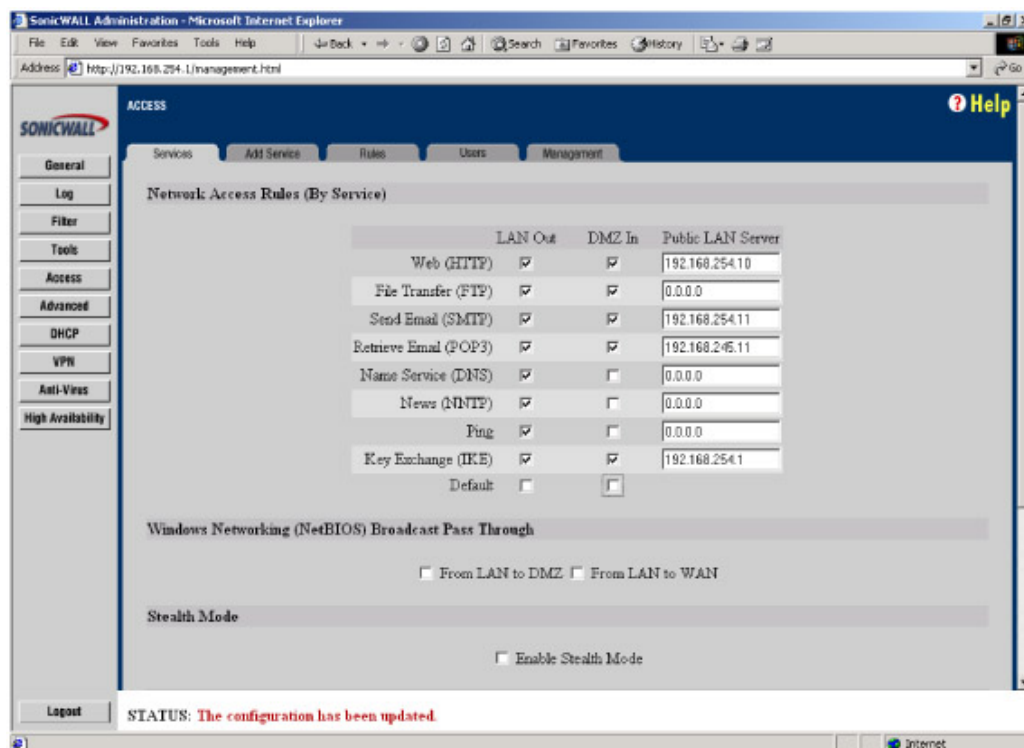


Figure 19.1: SonicWALL's web interface is the easiest to use that we've seen.

Security

A SonicWALL is a complete Layer-3 (Network layer) firewall. It does not do Application-layer proxying or content filtering. It has a simple HTTP filter included that can strip Java, ActiveX, and cookies, but no more than that. Its packet filter, port blocking and redirection, and VPN configuration are first rate and easy to configure.

Cost and Support

SonicWALL is neither cheap nor expensive, but when you add up the hardware and software costs for anything but a free-software firewall (see Chapter 16), the SonicWALL is very competitive in price. And if you instead add up the time and effort needed to configure a free-software firewall, you'll most likely find that SonicWALL is still comparatively cheap. SonicWALL's technical support is a little anemic, but there's not much to go wrong with the device anyway.

The devices range in price from about \$400 for the SOHO small 10-user devices to around \$3000 for the PRO VX (which is the most useful and should be considered the baseline device for protecting a real network), all the way up to \$27,000 for the top-of-the-line SonicWALL GX 650.

One thing to keep in mind at the time of this writing: the Client VPN licenses for Sonic-WALL cost around \$70 each, and the VPN upgrade for the SOHO and XPRS firewalls (to enable the VPN connectivity) is also around \$500. The PRO devices and up all come with VPN enabled.

One nice thing about SonicWALL that distinguishes it from the WatchGuard firebox (see later section in this chapter) is that the SonicWALL firewalls are essentially the same in configuration and use from the bottom of the line (the SOHO units) all the way up to the top-of-the-line GX 650. They merely add a few features and use faster hardware as you go up the product line. The interface is the same from box to box. The smallest Watch-Guard (the FireBox SOHO) is really a completely different device from the excellent Fire-Box 1000 and is configured and interfaced to separately (via the Web instead of by a Windows client application).

WatchGuard Firebox 1000

If you want a full-featured proxying firewall that doesn't take a rocket scientist to set up, the WatchGuard Firebox may be just what you're looking for. This product vies with the SonicWALL in price, capabilities, and ease of use, and just by looking at the two firewalls it's obvious that they're fighting over the same market segment. Of the two, the SonicWALL is easier to configure (requiring only a web browser on a client inside the network), while the WatchGuard includes support for proxying and content filtering that the SonicWALL does not.

•	
Pros	Cons
No hardware or software required	Can only be managed from Windows clients
Strong Application-layer inspection	
Strongest device-based firewall	
Highly reliable	

We had to scrape to come up with a negative for the above table—this device functions exactly as a theoretically perfect firewall would. It contains no significant failure components so it's reliable, yet it performs strong Application-layer filtering and is easy to administer. The interface isn't quite as easy as the SonicWALL devices, but it allows you to perform real-time monitoring that the SonicWALL can't. And when you consider that these devices cost about the same, they're the firewall of choice for higher security environments with more experienced staff.

Major Feature Set

The Firebox 1000 provides the following major features:

- Packet filter (stateful)
- Network Address Translator (dynamic, static)
- DMZ support
- Port redirection
- Proxies (DCE-RPC, FTP, H323, HTTP, RealNetworks, RTSP, SMTP, Stream-Works, VDOLive)
- Secure authentication (Proprietary, Windows NT, RADIUS, SecurID, and CRYPTOCARD)
- VPN (proprietary, DES, 3DES, IPSec/IKE, PPTP)
- VPN client software (Windows 98/NT/2000/XP, Unix, Linux)
- Bandwidth control and quality of service
- Logging and e-mail notification

The most impressive aspect of the Firebox 1000 is its built-in proxy support, a feature not found in other device-based firewalls (i.e., firewalls that don't expose you to the underlying operating system). Its VPN support, network address translation, packet filtering, and DMZ support are all first rate, but the same could be said of most other firewalls of its class. VPN support, which just a couple of years ago was a novelty in a device-based firewall, is now the order of the day—certainly in the future everybody's "drop in firewall" will have built-in proxying, but if you want it now and you want it easy to use, the Firebox 1000 is pretty much it.

Minor Feature Set

This firewall supports the following minor features:

- Network transparent drop-in configuration
- Content filtering (Java, virus scanning, URL blocking)
- Scan detection, spoofing detection, and automatic blocking
- DHCP
- Graphical administration
- Remote administration
- Centralized administration
- SYN-flood protection
- Anti-spoofing control
- Real-time monitoring and reporting
- Policy-based configuration and management
- High performance

Proxying is only half of securing ports for Application-layer protocols like HTTP, SMTP, and FTP. Proxying is important because it makes sure that the ports are being used for the protocols they were meant for, but it does not protect interior computers from malicious content (such as devious ActiveX controls and viruses) that are sent via those protocols. Content filtering is the other half of securing the ports, and the Firebox does that as well.

The firebox is also good at incident detection—telling you when you're under attack (and what kind of attack you're facing). The real-time graphical monitor is nice to watch—you can see traffic pattern changes as they happen. The lights on the front of the box are also helpful and intuitive: it is obvious at a glance how much traffic is flowing to or from the DMZ and the Internet, the protected LAN and the Internet, or between the DMZ and the protected LAN.

A nice feature of the WatchGuard 1000 firewall is that if you already have a publicly routed subnet that you want to protect, then you can place the firewall in "drop in" mode—where it is given an IP address on that subnet (rather than being set up as a router for that subnet), and it transparently intercepts the traffic between that subnet and the Internet. You have to place it connection-wise between the subnet and the router, but you don't have to reconfigure the clients or the router to protect your LAN.

Installation

After installing a number of command-line based free firewalls (see Chapter 16) and firewalls that run on top of Unix or Windows (see Chapters 17 and 18), installing and configuring the Firebox 1000 was a breath of fresh air. The graphical Windows application for administration was a breeze to install and use. After setting the IP addresses of its interfaces and giving it a range to supply for DHCP, the box was ready to use in a minimally configured state.

Security

A Firebox 1000 that is fully locked down with proxies in place is about as secure as you're going to get with a modern firewall. Perhaps OpenBSD does a better job of obfuscating TCP sequence numbers, perhaps Gauntlet has a better set of proxy services, but for the price and ease of use there's no comparison. Because the Firebox is based on Linux, its TCP sequence number generator is considerably more random than most devices.

Interface

The Windows client application that comes with the firewall for administration is easy to set up and use. The only easier way to administer a firewall is through your web browser (SonicWALL does this, as do the majority of the little home-office firewalls), because the management application limits you to configuring the machine from Windows (as opposed to, say, Solaris). See Figure 19.2 for a view of the Firebox management interface.

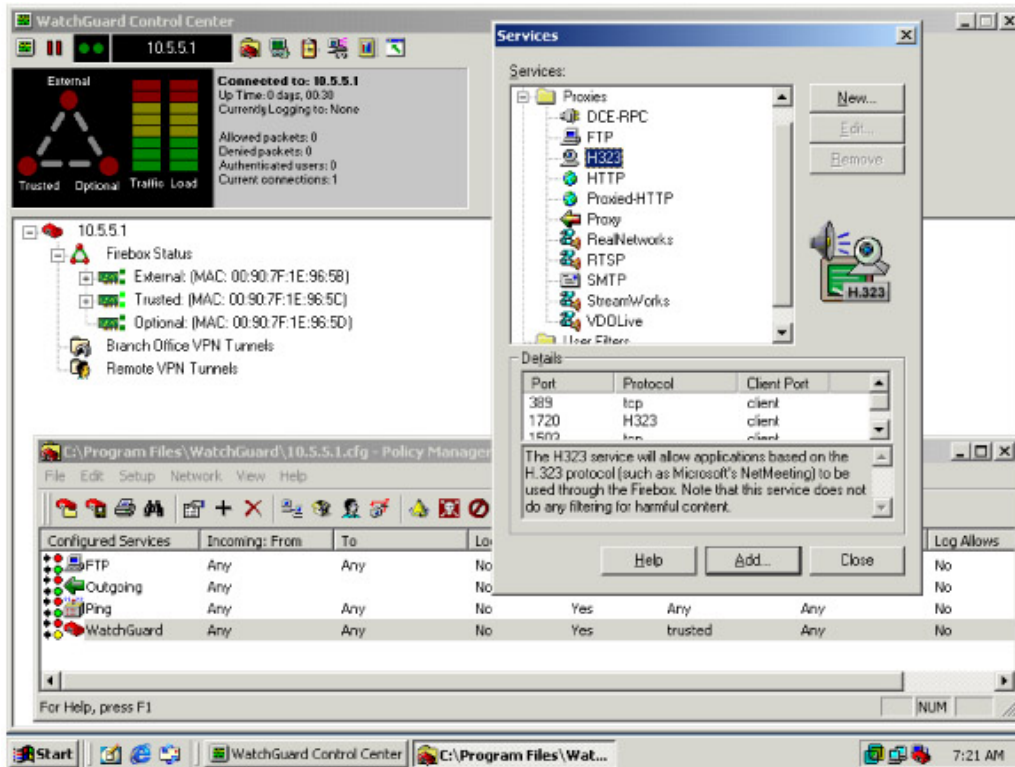


Figure 19.2: Firebox's rule-based interface

The Windows application does have the advantage that you can do more from it, including real-time monitoring of the status of the firewall. The policy-based rule editor is also easy to use, including allowing you to save a policy locally before uploading (so you can test out new configurations, for example, and fall back if they're too restrictive).

Documentation

The installation booklet provided with the firewall concisely and clearly walks you through the process of installing the firewall, but you'll have to look to the documentation supplied on the CD in PDF format for instructions on how to make policies to really secure your network.

The PDF documentation walks you step-by-step through using every feature of the Firebox, including establishing policies, setting up VPNs to other Fireboxes and to remote Windows clients, blocking URLs, and setting up content filters. It doesn't go into great detail explaining why you would do any of these things, but another book (such as this one) can tell you what to do to protect your network; the Firebox documentation will tell you how to do it.

Cost and Support

A WatchGuard Firebox is not cheap; at the time of this writing the Firebox 1000 will cost you about \$3000. Getting the top-of-the-line model (a model 4500) can cost \$7700. The support is good

though, including (in addition to your regular dial-up support) online documentation, questions and answers, and a web-based forum on which customers can exchange problems and solutions.

The home unit, which is really a different device entirely but can be used to establish a VPN connection to a model 1000, costs about \$300, though the VPN upgrade for it costs another \$400.

Elron Firewall

Elron Firewall is available on its own proprietary operating system and was ported to Windows NT in its latest edition. I find the port to NT interesting in light of the fact that Elron considers their secure OS to be one of the primary features of their firewall.

-

Pros

Fast stateful inspector firewall

Includes VPN

Supports IPX

Minimal hardware

Cons

No proxy servers

Adapters limited to 3c905

Ethernet

Poor user interface design

Elron employs multilayer stateful inspection rather than proxy servers for filtering in the Application layer. This is somewhat similar to Firewall-1's support for HTTP and FTP filtering. Filtering in the Application layer is capable of blocking numerous attacks, but filters may not recognize certain attacks that proxies would not forward because the attack would not be created. In other words, filtering still passes the originally formed packet, so undetected malformations can still be routed through. Multilayer filtering is considerably more secure than Network layer filtering alone, but not as secure as security Application-layer proxies.

Elron Firewall running on its own operating system is not subject to standard operating system vulnerabilities. Although a proprietary operating system is not necessarily more secure than a standard operating system, few hackers attempt hacks against operating systems that are not widely deployed, so the firewall is not vulnerable to most of the exploits developed by hackers. Since superfluous firewalling services (like file and print sharing) are not provided, no holes exist in the operating system.

Elron software maintains that, because 32OS source code has not been released to the public, there is virtually no possibility that hackers will be familiar with it. While this may be true to some extent, good hackers can read machine language source code through a process called *disassembly*, where the binary image is turned back into human-readable assembly language. While assembly language is not nearly as clear as the C programming language (relatively speaking), hackers who are familiar with the i386 microprocessor and its descendants could read it and thereby understand in detail the operation of a piece of proprietary software. I've done it, and so can any decent programmer. Though software based on a proprietary operating system will keep the masses at bay, security through obscurity should never be relied upon. Note also that 32OS uses MS-DOS as a boot loader, and could therefore be susceptible to certain types of RAM resident viruses.

Elron's documentation describes some alarming problems that can happen when the firewall runs out of memory, including losing Network Address Translation addresses, which would cause translated connections to be lost. While neither fatal nor a security risk, these sorts of problems are

the result of using proprietary operating systems that aren't completely thought out.

Hardware requirements for the Elron Firewall are (SecureOS Version):

Connections <1.5Mb/sec (T1)

- 486DX-2/66
- 8MB RAM
- 200MB hard disk drive
- MS-DOS 6.22
- Two or three 3C905 10/100 NICs
- Floppy drive

Connections >T1

- Fastest possible processor
- 16MB RAM

Requirements for the management station are:

- Windows 9x or NT
- 50MB available disk space
- 16MB RAM

Major Feature Set

Elron Firewall provides the following major features:

- Stateful inspection packet filter
- Network Address Translation
- Encrypted authentication
- Virtual Private Networking

Elron Firewall's stateful inspection filter is unique in that it is capable of filtering the application (payload) portion of a packet for known content. The firewall compares packets to bit-patterns of previously filtered packets before passing the packet into the protected network. This ensures that unknown deformations of packets will be filtered out.

Elron Firewall's NAT option supports IP address hiding only by using the Firewall's IP address. This provides an upper limit of about 64,000 outbound connections, but that's generally high enough that this limitation is not serious for most organizations.

User authentication clients are provided for Windows 9x and NT. Authentication is password-based and supports RADIUS and CHAP authentication. The user authentication software also supports periodic authentication.

The included VPN option provides IP in IP tunneling, which provides a measure of internal security by hiding the true source and destination addresses. IPSec is used to encrypt the encapsulated IP packet.

Elron makes two completely separate Application layer filters called the InternetManager (HTTP) and the MessageInspector (e-mail, news, and FTP). These products run on their own Windows NT

server and work with any firewall or security service. The Message–Inspector filter performs powerful keyword string matching and statistical analysis (for spam filtering) to block e–mail, newsgroups, and FTP downloads.

Minor Feature Set

Elron supports the following noteworthy minor features:

- IP and IPX filtering
- VPN continuous key regeneration

Elron supports both IP and IPX filtering. IPX filtering is not usually a big concern unless you run a large IPX network where internal security between divisions is important. For most enterprises, IPX filtering is not a function required of bastion hosts. The firewall also supports IPX bridging (forwarding all IPX packets transparently and irrespective of their contents), which is not a security function and reduces the security posture of your network.

The continuous key regeneration feature provides a facility somewhat akin to Kerberos ticketing. After an established amount of VPN traffic has passed between two firewalls, the firewalls will both generate new keys and exchange them. This reduces the amount of useful time a brute–force–decrypted key would be useful, thus moving the probability domain for a brute–force attack from highly unlikely to practically impossible.

Interface

Elron firewall is configured remotely through a Windows–based policy manager. The firewall itself is initially configured using the firewall management software on a Windows computer and transmitted to the firewall located on the same Ethernet collision domain.

The user interface bespeaks an amateurish attempt at design, suffering from such problems as a non–sizeable main window that takes up the entire screen and the use of purely modal dialogs throughout the software, which prevents you from seeing two content windows at the same time. There seems to be an unwritten rule in the firewall industry that user interfaces aren't worthy of programming effort. Figure 19.3 shows the clunky management interface.

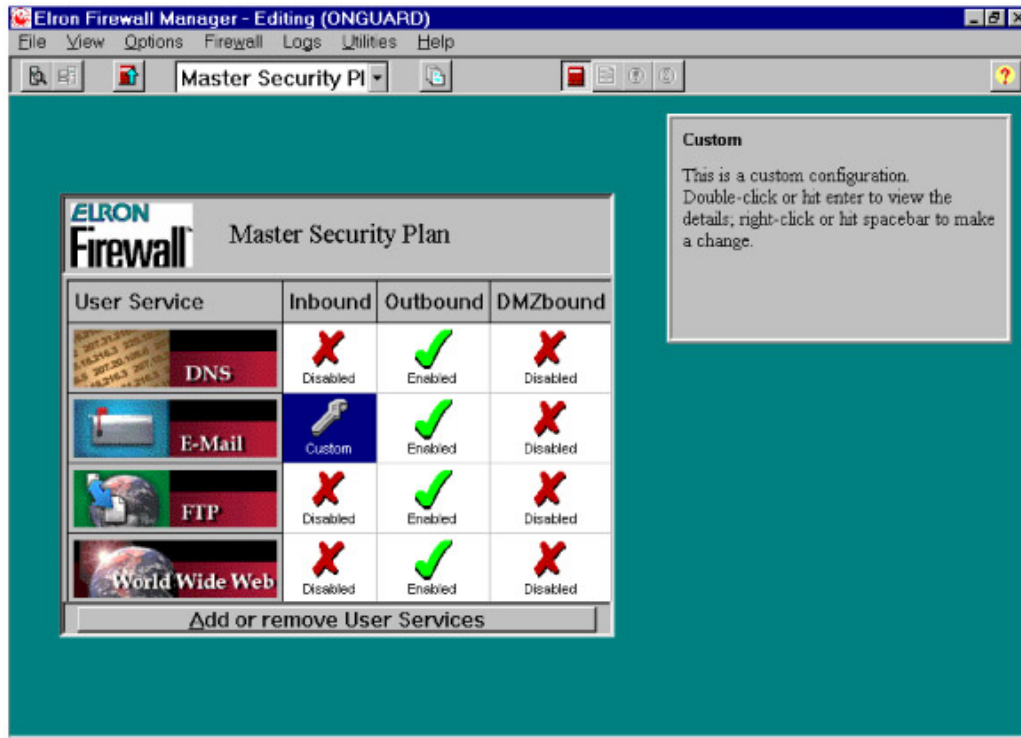


Figure 19.3: The Elron Firewall Management Interface

The interface is not particularly easy to use since it doesn't conform to any specific interface methodology. In some cases, you right-click to access features, while in others you double-click. There's also no indication of which interface elements can be activated and which can't.

Security

Elron Firewall's multilayer inspection filter is the heart of the firewall. The multilayer filter is interesting because it can filter content in the Application layer to reject unrecognized information. The level to which this functionality is actually used varies from protocol to protocol, but administrators can customize it on a per-protocol basis.

Customizing the firewall is not easy and requires a solid knowledge of TCP/IP and firewalling. If you've read through this book so far, you'll have no problems.

Elron Firewall running on 32OS should be considered hardened, since no OS-specific exploits are known for the operating system. This makes it equivalent at least to a Windows NT installation with no extra services running, no extra user accounts, and in a state of complete lockdown where additional software (like Trojan horses) cannot be installed.

Hardened operating systems are operating systems in which no extraneous services or exploitable mechanisms exist because the operating system simply doesn't support them, or in which all software functions not directly related to supporting the firewall have been disabled. Hardening an OS reduces considerably the number of vectors a hacker can attempt to exploit, and thereby dramatically improves the security posture of the firewall system.

Documentation, Cost, and Support

Documentation is provided in PDF format and is very strong. It is highly task oriented, containing detailed procedures for performing most firewall administrative tasks, yet adequately covers the theory behind the features used. Most administrators will have no trouble getting the firewall up

using the supplied documentation.

Elron has a small searchable knowledgebase online. Technical support is available via e-mail. Elron is a wholly owned U.S. subsidiary of the Israel-based Elron Electronic Industries.

Elron Firewall comes in various user levels and prices:

- 25 users: \$1,000
- 50 users: \$2,000
- 100 users: \$3,250
- 255 users: \$5,600
- Unlimited users: \$9,000

Elron also sells annual maintenance contracts at about 25% of the initial purchase price.

GNAT Box

GNAT Box is an ICSA certified stateful inspection packet filter and Network Address Translator that runs on its own operating system, which it boots from a single floppy disk. GNAT Box also includes an SMTP proxy and a split DNS server. GNAT Box protects against IP spoofing and common denial-of-service attacks. Although GNAT Box does not provide a Unix operating system environment, its kernel and TCP/IP stack are derived from BSD, which is an open source Unix operating system. You can think of GNAT Box as a highly optimized firewall-specific distribution of the BSD operating system.

Tip GNAT Box can now be purchased on a 16MB Flash device that plugs directly into the IDE connector on the computer's motherboard to replace the disk drive. This "devicefies" the firewall by removing the primary failure component. Plus it's really cool, if you like that sort of thing.

•

Pros

Runs on cheap hardware
No standard OS security holes
Fast
Inexpensive—less than \$1,000 for unlimited use

Cons

No user authentication
Cannot integrate with third-party tools
No content scanning or additional proxies

The complete system requirements are:

- 386 or higher Intel-compatible microprocessor
- 8MB RAM, 16MB recommended for e-mail proxy, more than 32MB is not useful
- Floppy disk drive
- Two network adapters
- Display adapter
- Printer port (to attach a copy protection key circuit)

As you can see, the requirements are quite minimal—you won't even need a hard disk drive, and nearly any obsolete PC you have lying around will work fine. The big drawback to GNAT Box is its limited support for network adapters. Because the operating system is based on BSD, only network adapter drivers for that operating system are available. It appears that the vendor actively usurps

open source adapter drivers for its firewall. So, many popular Ethernet models are available, but you can forget about using adapters that are even slightly esoteric. Support is provided for most adapters from:

- 3Com
- Compaq
- DEC
- SMC
- Intel (except ISA bus adapters)
- Various others based on similar chipsets

A complete list is provided with the documentation. If you have a problem getting your adapters to work, you will need to purchase adapters from the supported list. This is usually not a problem, since most of the adapters are available for well under \$100 each.

GNAT Box performs faster than most Internet connections. Performance is limited primarily by the speed of the network adapters, so the performance of GNAT Box is at least as good as the fastest PC-based firewalls. GNAT Box does not support VPN-encrypted tunnels or remote user authentication.

You'll find that GNAT Box devices have a hard time dealing with multiple inbound connection attempts at rates higher than 1.5MB. The software is optimized for protecting office networks with minimal interior services, not for protecting a bank of public servers.

Tip A downloadable evaluation edition of GNAT Box is available at <http://www.gnatbox.com/>.

Major Feature Set

The following major features are included in GNAT Box:

- Stateful inspection packet filter
- Network Address Translator
- DMZ support
- Port redirection
- VPN (AES, DES, 3DES, IPSec/IKE)
- VPN client software (PPTP, IPSec, SSH)
- Firewall high availability (1000GB)
- Logging and e-mail notification
- E-mail and HTTP Proxy

GNAT Box's stateful inspector is fairly sophisticated. By detecting "hard-to-firewall" protocols like FTP and real-time multimedia protocols on the way out of your network, the firewall will create virtual cracks, a term GNAT Box uses to describe temporary holes created for the return channels of these protocols. This makes GNAT Box compatible with these protocols without compromising security by simply opening up permanent holes.

The e-mail proxy receives and regenerates e-mail messages. It includes light anti-spamming features, but does not include content filtering or attachment blocking. The built-in web proxy filters URLs; for a more sophisticated HTTP filter it can redirect web requests to an external HTTP proxy.

Minor Feature Set

GNAT Box provides the following minor feature:

- Demand dialing of PPP connections

If configured to use a PPP connection as the external interface, GNAT Box will automatically dial the interface on demand whenever an internal client requests Internet data. The PPP interface can also be set to dedicated (dial on boot) or manual-enable mode (whenever the administrator enables the link).

Interface

The firewall itself has a text-based console interface, but you need never use that interface unless you want to administer the box directly without a management workstation. System administration is performed on an administration computer that connects to the GNAT Box via a TCP port you define. Figure 19.4 shows the GBAAdmin utility's interface.

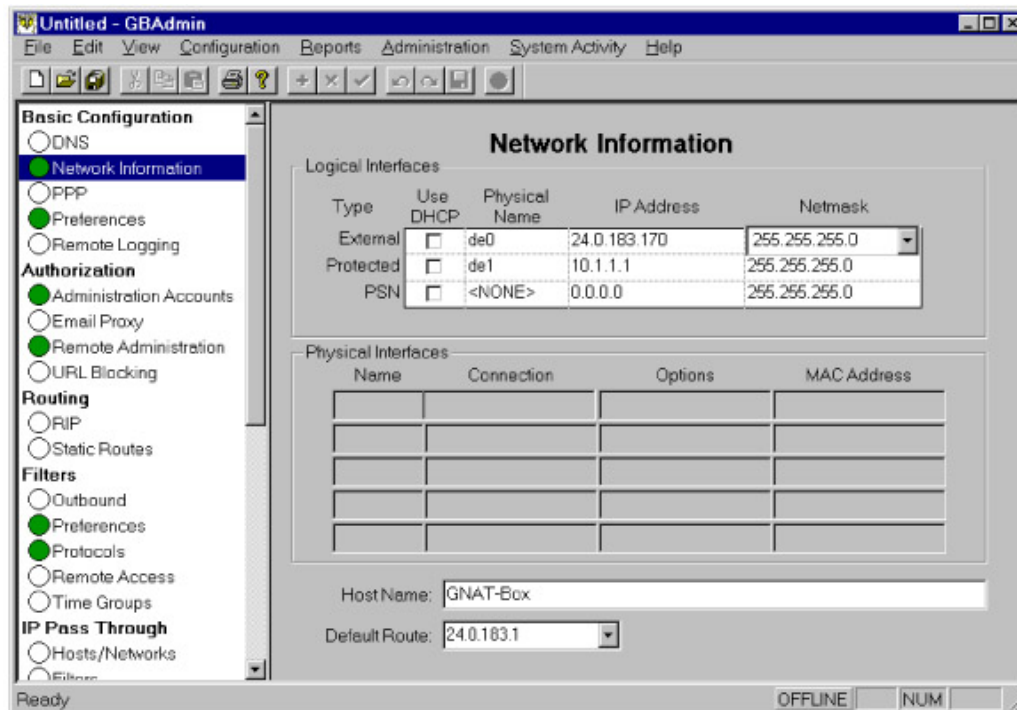


Figure 19.4: The GNAT Box administration interface

The GBAAdmin utility requires Internet Explorer 3.0 or higher in order to operate correctly.

You can use the GBAAdmin utility to configure an existing GNAT Box over your network, or to create a combined configuration and runtime floppy from which a fully configured GNAT Box will boot.

Security

Because GNAT Box is absolutely the only software running on the firewall platform, it should be considered more hardened than any firewall running on a standard operating system. There's no way to run an arbitrary program on a GNAT Box firewall, so hackers cannot directly exploit it. There are no extraneous services to exploit and no extra information leaking services that a hacker might find useful. The same machine cannot be used to support mail, web, or any other services. GNAT

Box turns a PC into a firewall appliance and achieves a high degree of inherent security that way.

Documentation, Cost, and Support

Documentation consists of a PDF user's manual that is both extremely well written and very clear. Firewalling concepts are explained in detail.

GNAT Box is free for home and non-commercial use, for five or fewer users. The full version costs \$800 for unlimited users. No other pricing is available. Technical support is available via the Web at <http://www.gnatbox.com/> or via e-mail.

BorderManager

BorderManager is an ICSA-certified suite of services that run on Novell NetWare servers; they provide firewalling, Network Address Translation, proxying, authentication, and VPN services. These three packages are available separately or bundled together in the Enterprise edition. BorderManager runs on NetWare 5 and NetWare 4.x platforms.

•	
Pros	Cons
Runs on NetWare	No unlimited use license
Integrated with Novell NDS	Content filtering is proprietary and not well supported
SOCKS access for IPX clients	Clunky setup and interface
Strong remote management	Components not integrated

Because NetWare is highly optimized for speed, BorderManager's minimum requirements are lower than those of most full-spectrum firewalls. The platform requirements are:

- Intel i486 or higher microprocessor
- 32MB RAM (128MB recommended)
- 500MB hard disk drive, additional for caching
- Novell NetWare 4.x or 5
- At least two network interfaces

Major Feature Set

BorderManager provides a full suite of high-end firewall features:

- Stateless packet filter
- Security proxies for numerous protocols
- Network Address Translator
- Secure authentication
- Virtual Private Network

The weak point is the stateless packet filter, but since all protocols run through security proxies, the packet filter need only filter for denial-of-service attacks, which are far less common on NetWare than NT or Unix for lack of hacking effort against the platform.

BorderManager includes a very complete set of security proxies, including proxies for real-time

multimedia services. Most proxies require SOCKS-enabled clients, however, bespeaking a lack of integration with the NAT component of the firewall, and reliance upon a basic SOCKS proxy as the source for many of the application proxies.

The Network Address Translator actually works at the Application Layer as something of a generic TCP proxy. Because it can be made transparent, it's something of a middle ground between a true NAT and a generic TCP or SOCKS proxy.

Secure authentication is provided by the BorderManager Authentication Services component, which is included in the enterprise edition of the firewall. BMAS is essentially an NDS plug-in for RADIUS authentication protocol.

VPN support is provided by the BorderManager VPN Services component, which is included in the enterprise edition of the firewall.

Minor Feature Set

BorderManager supports the following minor features:

- Java content filtering
- Logging to databases
- Reverse proxies for HTTP and FTP
- SOCKS support for IPX clients
- Dial-up service software

BorderManager's HTTP proxy supports Java content filtering, but strangely lacks support for ActiveX filtering, an application protocol that is far more dangerous.

BorderManager's logging facility allows you to connect to ODBC-compliant databases. If you've ever tried to export and import a massive firewall log to search for specific attacks, you'll really appreciate a direct database connector. As with all firewalls, you'll have to supply your own database software.

BorderManager supports a strange form of reverse-proxying of external connections to internal (or DMZ) web servers. Reverse proxy usually refers to the provision of multiple web servers behind a single IP address, but Novell uses this term to refer to caching outbound web pages on the proxy server itself, thus eliminating second and subsequent accesses to the web server from public clients. This essentially makes the proxy server your web server (unless you have an interactive website), which is rather pointless since the proxy server isn't any faster at serving web pages than a good web server would be. The server will store web pages in its RAM cache for faster access, but Unix and NT web servers do that automatically as a file system function.

Like Microsoft Proxy Server, BorderManager is capable of proxying TCP/IP Application-layer protocols like HTTP and e-mail to internal clients that run on IPX. Using IPX as your interior network protocol makes network configuration faster, easier, and more secure since no direct route to clients exists, even if your firewall is compromised. The hacker would actually have to subvert the proxy functionality of the firewall to convert TCP/IP to TCP/IPX at the gateway in order to reach interior clients. This is practically impossible.

Novell Internet Access Server (NIAS) provides dial-up service akin to Windows NT's RAS service and is included in the BorderManager Enterprise Edition. NT and Unix both natively support dial-up, so this isn't a particularly compelling reason to choose this firewall over those based on more

common operating systems.

Interface

BorderManager runs on NetWare 4.x and 5 and therefore uses the clunky text-based console interface of NetWare servers. ClearView, a Windows-based monitoring tool for BorderManager, is included; so, once the software is set up and configured, you'll perform most of your administration from Windows-based workstations.

BorderManager integrates seamlessly into the NetWare's NDS directory, so you can select and manage firewalls and their options through the hierarchical browser NDS provides. NetWare administrators will find this compelling, but attempting to manage the firewall in non-NDS environments would not be compelling. For this reason, Border-Manager is a serious option only in NetWare environments.

BorderManager is a large package consisting of numerous components that are installed and configured separately. You should be at least a Novell CNE and be very familiar with the NetWare environment before attempting to install this software.

Security

Proxies are available for the following protocols:

- HTTP and SSL
- FTP
- DNS
- Gopher
- SMTP and POP3
- NNTP
- RealAudio and Real Video
- Real Time Streaming Protocol (RTSP)
- SOCKS 4 and 5
- Generic TCP/UDP
- Telnet

Most of these proxy applications are merely configurations for a generic SOCKS proxy, which means that the clients have to support connection to a proxy server. Only the HTTP proxy can operate in transparent mode so clients do not need to be configured for proxy operation.

Transparent proxy operation refers to the ability of a proxy to operate without being specifically configured on each client. Rather than being addressed on a certain port (usually 8080 for HTTP proxies) to which all web browsers must be configured, transparent proxies (since they also act as Network-layer routers) inspect the routed traffic going from the internal to the external network and detect HTTP traffic on its way out. They then transfer these HTTP requests to the proxy service rather than forwarding them directly, and thereby insert the proxy functionality seamlessly and transparently. This makes it impossible for internal clients to bypass the proxy and eliminates the administrative burden of configuring the proxy.

Documentation, Cost, and Support

The documentation for BorderManager is clearly written, although relatively shallow in concepts and theory. The various topics are given considerable detail and are task oriented, so firewall

administrators should be able to establish and configure the firewall without problems.

Novell's posted comparison matrix on their website is laughable and filled with misleading information about their competitors' products. According to this bizarre document:

- Microsoft Proxy Server supports Network Address Translation. (In fact, it doesn't operate below the Application layer.)
- Firewall-1 supports only static packet filtering. (In fact, Checkpoint was first to market with a stateful inspection firewall.)
- Firewall-1 supports content proxying. (In fact, only the SMTP service in Firewall-1 could be called a proxy. The other services are Application-layer content filters.)
- BorderManager supports stateful packet inspection. (In fact, BorderManager relies upon Application-layer proxies to maintain connection information.)
- Base price for BorderManager is listed as \$995. (In fact, you also have to pay a peruser licensing fee that makes BorderManager very expensive indeed.) Novell compared this to MS-Proxy Server's all-inclusive price and Firewall-1's price for a 25-user version.

Fortunately, most of the rest of Novell's documentation appears to be written by people who knew what they were talking about.

Pricing is simple: \$1,000 for the BorderManager Firewall Services, plus \$30 per user, no volume discounts. License packs are available in 10, 25, 50, 100, 250, and 500 user increments. BorderManager includes a 2-user version of NetWare upon which the software is installed.

Technical support is available via Novell's extensive website at <http://www.support.novell.com/>, which includes a very thorough searchable KnowledgeBase. You may also receive support from one of the hundreds of thousands of consulting Certified NetWare Engineers throughout the world.

IBM Firewall for AS/400

IBM's family of firewalls is based on the firewall technology that IBM has been developing for 15 years. IBM has firewalls for AIX/6000 on RS/6000 microcomputers, Windows NT on Intel microcomputers, OS/400 on AS/400 minicomputers, and OS/390 on 390 series mainframe computers. Interestingly, IBM hasn't bothered to port its firewall to its own OS/2 operating system. This section discusses the latest version of IBM's Firewall for the AS/400.

•	
Pros	Cons
Runs on esoteric IBM platforms	Expensive
Performs OS hardening	Poorly integrated
Firewall runs on embedded separate PC processor	Lacking true content filters or strong security proxies
	Stateless packet filter
	No support for DMZ

The AS/400 firewall actually runs on an integrated PC server embedded in the AS/400 computer. This provides system separation so that in the event that the firewall is compromised, no access to applications running on the AS/400 is achieved. The software runs from a read-only hard disk drive, so the firewall cannot be modified once it's installed.

IBM's firewall is something of an "also-ran" in the firewall field. It supports most of the major technologies, but it suffers from poor integration, the requirement for third-party utilities for numerous security functions like alerting and scan detection, and a lackluster effort in the development of secure proxies for major TCP/IP protocols.

IBM Firewall for AS/400 requires:

- OS/400 version 4 release 1
- TCP/IP connectivity utilities for AS/400
- Integration services for FSIOP
- RISC AS/400 with integrated PC server

Integrated PC server must have:

- 32MB RAM (64 recommended)
- 486 or higher microprocessor

Major Feature Set

The major feature set for the IBM Firewall for AS/400 includes:

- Packet filter
- Network Address Translator
- Proxies for HTTP, SMTP
- SOCKS proxy

The packet filter is simple and stateless. It provides no functionality beyond the functionality provided by IPChains or NT's built-in packet filtering. Services can be either passed or blocked based on their TCP port number, and the filter can use the ACK bit to deny inbound connection attempts.

The firewall is not normally configured to forward IP packets; rather, outbound connections are achieved via the circuit level gateway (i.e., SOCKS proxy) running at the Application layer. This means that client software incompatible with SOCKS either cannot be used or relies upon the enabling of IP forwarding, which defeats many of the security features of the firewall. Support for real-time streaming multimedia protocols like Real Audio and H.323 also requires enabling of packet forwarding.

Security proxy services are provided for HTTP and SMTP. All other TCP services must be SOCKS compatible in order to work with the firewall as its remaining functionality is provided by a SOCKS circuit level gateway.

Minor Feature Set

IBM Firewall for AS/400 provides the following minor feature:

- Installs as standard AS/400 application

The IBM Firewall for AS/400 installs as a normal AS/400 application, so AS/400 operators will be familiar with its operation.

Interface

Firewall administration is performed through a web browser using an HTML-based administration tool. The tool is simple, but it provides an adequate interface to the firewall. Very little policy abstraction exists, so a strong knowledge of TCP/IP is required. Figure 19.5 shows the configuration interface running in a web browser.

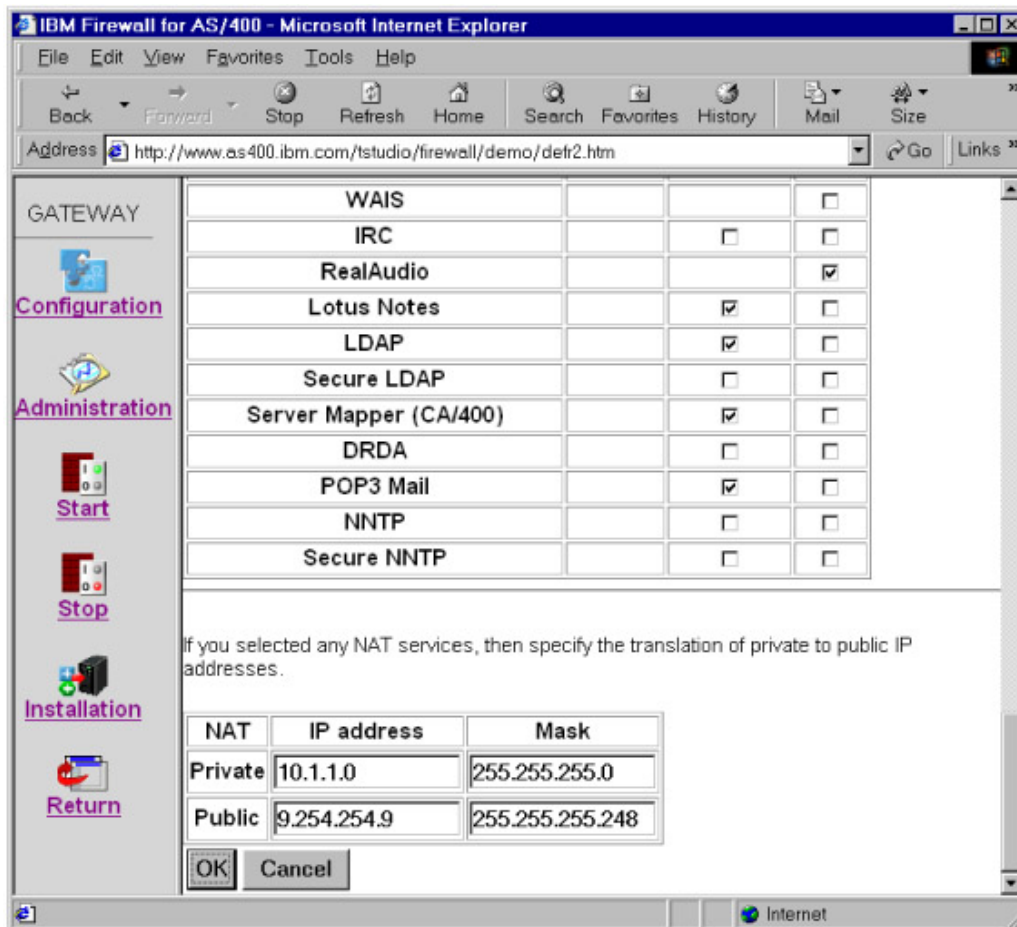


Figure 19.5: The IBM Firewall for AS/400 Management Interface

Security

The IBM Firewall for AS/400 relies primarily upon its SOCKS proxy for security. The filter is stateless and suitable primarily for protection against denial-of-service attacks. Network Address Translation is achieved via the SOCKS mechanism. True security proxies are available only for HTTP and SMTP.

The main AS/400 processor can disable the firewall if tampering is detected (the documentation does not specify what sort of tampering is detectable).

Firewall uses an implicit "deny all services" policy, so every service that is enabled must be explicitly enabled by the firewall policy. Remote administration is disabled by default, but can be enabled by the administrator.

In sum, the firewall is rather dated in its implementation and runs on esoteric expensive hardware. You will achieve better security at lower cost by using a non-integrated firewall.

Documentation, Cost, and Support

Documentation is available in hard copy or PDF format. The documentation is simplistic and lacking in real implementation detail, but it is task oriented and will get you through the configuration.

Support is available via IBM's direct consulting services or through IBM consulting resellers. Pricing information for OS/400 was not available; prices for the AIX version are as follows:

- 25 users: \$2,499
- 50 users: \$4,499
- 250 users: \$9,499
- Unlimited users: \$16,500

Case Study: Drop 'Em in, Turn 'Em on Firewalls

The new device-based firewalls that are taking over the firewall market are taking over for a reason—they are easy to install and use. The more difficult a firewall is to configure, the easier it is to get the configuration wrong. We've been using device-based firewalls in all our client's networks because our time is valuable to us and to our customers, so the less time it takes us to set things up, or (more importantly) to fix things, the happier our customers are with us.

Firewalls like Elron and GNAT box hold down a middle ground between computer-based firewalls and device-based firewalls; they run on standard i386 hardware, but they package their own OS with the firewall and are managed like a device. These firewalls, while interesting, are also losing ground against the device market because you still have to provide a computer, which means that they only sell to businesses that have IT staff with excess time for building computers.

But other firewalls like BorderManager and IBM Firewall for AS/400 exist more as a way to round out the vendors' product offerings and catch the business of the occasional corporate buyer who is purchasing based on a checklist than to provide cost-effective strong security. These offerings are sold to shops that have a fairly blind loyalty to their vendors or to consultants who have only a single operating system expertise support. These firewalls can't really compete with the stronger and cheaper firewalls available for more common operating systems and to those used as firewall embedded devices. This explains why you don't see them around much. Since the future of any product offering is ensured by its success, you'll probably find that these firewalls fade away as the more competitive Unix, Windows, and dedicated hardware firewalls carve up the market. Their vendors probably won't be able to justify the continual improvement necessary to remain competitive in the war against hacking.

Even my dyed-in-the-wool loyal IBM customers who use AS/400s opt for other firewalls: Cisco PIX in one installation I support and Firewall-1 in another. All the NetWare installations I'm involved with use NT-based firewalls.

List of Figures

Chapter 1: Understanding Firewalls

- Figure 1.1: Filtered Internet connections block undesired traffic.
- Figure 1.2: Proxy servers receive requests on the private network and regenerate them on the public network.
- Figure 1.3: Filtered packet service
- Figure 1.4: A single firewall with public servers exposed to the Internet
- Figure 1.5: A single firewall with public servers protected but allowing external traffic in through the firewall
- Figure 1.6: Two firewalls acting in concert to completely protect a network
- Figure 1.7: A DMZ firewall provides different security for different needs.
- Figure 1.8: Multiple firewalls in an enterprise
- Figure 1.9: The disconnected security model provides the most protection from Internet intrusion.

Chapter 3: TCP/IP from a Security Viewpoint

- Figure 3.1: The TCP/IP protocol suite is composed of layers of services that roughly correspond to the layers of services defined in the OSI network model.
- Figure 3.2: Leased line data rates range from 56Kbps all the way up to 2.5Gbps.
- Figure 3.3: The structure of Ethernet and PPP frames are tailored to their uses (Ethernet for fast shared LANs, PPP for slow dial-up links).
- Figure 3.4: Each layer of the OSI stack adds layer-specific data to what it receives and passes the expanded information to the layer below it. When the layer receives information from the layer below it, the layer removes layer-specific data and passes the information on to the layer above it.
- Figure 3.5: An IP packet has a header that includes the source and destination IP addresses, version, type, and service information, options, and a data section.
- Figure 3.6: The subnet mask identifies which part of the IP address is the network address and which part is the station address.
- Figure 3.7: A gateway machine has two network adapters and allows network traffic to move between LANs.
- Figure 3.8: An ARP/RARP packet is a broadcast request for data.
- Figure 3.9: The routing table instructs the gateway how to forward IP packets.
- Figure 3.10: A gateway should know how to explicitly route to any internal network and also have a default external gateway to send packets it doesn't know what to do with otherwise.
- Figure 3.11: CMP messages share a common header format with Type, Code, and Checksum fields.

Chapter 4: Sockets and Services from a Security Point of View

- Figure 4.1: The DNS system distributes control of Internet domain names.
- Figure 4.2: A single DNS message may contain multiple address resolution queries or responses in the Question RR and Answer RR fields of the DNS UDP packet.

Chapter 5: Encryption

- Figure 5.1: In the Rot(n) substitution cipher, the plaintext is rotated several characters for encryption into ciphertext, and then rotated several characters back into plaintext for decryption.

Chapter 6: Packet Filtering

Figure 6.1: A packet filter rejects undesired traffic.

Figure 6.2: A stateful inspection packet filter allows return data.

Figure 6.3: A stateful inspection filter leaves "holes" open only as long as they're necessary.

Chapter 7: Network Address Translation

Figure 7.1: Network Address Translation

Figure 7.2: Using firewalls to perform load balancing

Figure 7.3: Network redundancy

Chapter 8: Application-Level Proxies

Figure 8.1: A service proxy

Figure 8.2: Load balancing with a proxy server

Figure 8.3: Transparent Proxies intercept and forward client requests without requiring client configuration.

Figure 8.4: Place proxies behind firewalls

Chapter 13: Security Utilities

Figure 13.1: The Windows Event Viewer showing the security log

Figure 13.2: The Windows Network Monitor

Figure 13.3: The Performance Monitor

Figure 13.4: The CACLS Permissions Tool

Figure 13.5: Tracert command sample trace

Figure 13.6: Results of a scan against a network

Figure 13.7: The ISS Internet Scanner

Figure 13.8: L0phtCrack decrypts passwords on an NT 4 Server.

Chapter 15: Windows as a Firewall

Figure 15.1: Windows NT Packet Filter Configuration

Chapter 16: Open Source Firewalls

Figure 16.1: IP Chains command-line interface

Figure 16.2: The Storm Linux graphical interface to IPChains

Figure 16.3: The Gnome graphical firewall administration (included in Red Hat Linux 7.2)

Figure 16.4: IPF takes its configuration from a text file.

Chapter 17: Windows Firewalls

Figure 17.1: Firewall-1's rule-based interface

Figure 17.2: Raptor Management Console for the Symantec Enterprise Firewall user interface

Figure 17.3: Microsoft ISA Server

Chapter 18: Unix Firewalls

Figure 18.1: The SunScreen Initial Policy page

Chapter 19: Device and Specialty Firewalls

Figure 19.1: SonicWALL's web interface is the easiest to use that we've seen.

Figure 19.2: Firebox's rule-based interface

Figure 19.3: The Elron Firewall Management Interface

Figure 19.4: The GNAT Box administration interface

Figure 19.5: The IBM Firewall for AS/400 Management Interface

List of Tables

Chapter 3: TCP/IP from a Security Viewpoint

Table 3.1: ICMP Message with general message type, specific message code, and whether the message is a query or a response to a network event

Chapter 7: Network Address Translation

Table 7.1: Firewall Routing Table Entries

Chapter 10: The Ideal Firewall

Table 10.1: ICMP Service Rules

Table 10.2: IP Service Rules

Table 10.3: UDP Service Rules

Table 10.4: TCP Service Rules

Table 10.5: The Additional Rules of a Cautious Administrator

Chapter 15: Windows as a Firewall

Table 15.1: Time required to crack common sets of passwords

List of Tables

Chapter 1: Understanding Firewalls

Case Study: Firewall Options

Chapter 2: Hackers

Case Study: The Metamorphosis

Chapter 3: TCP/IP from a Security Viewpoint

Case Study: Where Is the Data Coming From?

Chapter 4: Sockets and Services from a Security Point of View

Case Study: Leaving the Barn Door Open

Chapter 5: Encryption

Case Study: Encryption Messages

Chapter 6: Packet Filtering

Case Study: Leaky Filters

Chapter 7: Network Address Translation

Case Study: Trial by Firewall

Chapter 8: Application–Level Proxies

Case Study: Guilt by Association

Chapter 9: Virtual Private Networks

Case Study: VPNyone?

Chapter 10: The Ideal Firewall

Case Study: All The King's Firewalls

Chapter 11: Configuring a Real Firewall

Case Study: Scaling the Firewall

Chapter 12: Attack Profiles

Case Study: As You Sow, So Shall You Reap

Chapter 13: Security Utilities

Case Study: Hacking @Home

Chapter 14: Intrusion Detection

Case Study: Serious Intrusion

Chapter 15: Windows as a Firewall

Case Study: Windows as a Firewall?

Chapter 16: Open Source Firewalls

Case Study: Free vs. Commercial Firewalls

Chapter 17: Windows Firewalls

Case Study: Windows As a Firewall

Chapter 18: Unix Firewalls

Case Study: Try to Buy

Chapter 19: Device and Specialty Firewalls

Case Study: Drop 'Em in, Turn 'Em on Firewalls

List of Sidebars

Chapter 1: Understanding Firewalls

Variants of Windows

Chapter 2: Hackers

Reality Check: Ethical Hackers
Reality Check: Hacker Terminology
Reality Check: I Spy?
E-Mail from Heaven
Notes from the Front

Chapter 3: TCP/IP from a Security Viewpoint

How Layering Works
Network Byte Order (Big-Endian)
Class and Classless Addressing
ICMP Vulnerabilities

Chapter 4: Sockets and Services from a Security Point of View

Buffer Overruns and Denial-of-Service Floods
Service Separation
One Key Fits All
Reverse Lookup

Chapter 7: Network Address Translation

Reality Check: Forging E-Mail to Gain Control

Chapter 8: Application-Level Proxies

Sage Advice: Don't Make Me URL
Sage Advice: Beware the Open Filter

Chapter 9: Virtual Private Networks

Reality Check: Frame Relay vs. VPN
Reality Check: Business Velocity
Reality Check: The Importance of Using the Same ISP
Reality Check: 99% Reliability
Reality Check: A Tunneling Worm

Chapter 14: Intrusion Detection

Reality Check: Target or Opportunity?
Tricks of the Trade: Virtual Decoys

Chapter 15: Windows as a Firewall

NAT in Windows Workstations

Chapter 16: Open Source Firewalls

Example: IPChains Shell Script

Example: FWTK Configuration Entries

Example: Drawbridge Filter Source File

Typical IPRoute Configuration Scripts