

F O U R T H E D I T I O N



LINUX

FIREWALLS

ENHANCING SECURITY WITH NFTABLES AND BEYOND

S T E V E S U E H R I N G

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Linux[®] Firewalls

Fourth Edition

This page intentionally left blank

Linux[®] Firewalls

Enhancing Security with nftables and Beyond

Fourth Edition

Steve Suehring

◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Suehring, Steve.

Linux firewalls : enhancing security with nftables and beyond.—Fourth edition / Steve Suehring.
pages cm

Earlier ed. authored by Robert L. Ziegler.

Includes bibliographical references and index.

ISBN 978-0-13-400002-2 (pbk. : alk. paper)—ISBN 0-13-400002-1 (pbk. : alk. paper)

1. Computers—Access control. 2. Firewalls (Computer security) 3. Linux. 4. Operating systems (Computers) I. Title.

QA76.9.A25Z54 2015

005.8—dc2

2014043643

Copyright © 2015 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

Permission is granted to copy, distribute, and/or modify Figures 3.1 through 3.4 under the terms of the GNU Free Documentation License, Version 1.3, or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in Appendix D, “GNU Free Documentation License.”

ISBN-13: 978-0-13-400002-2

ISBN-10: 0-13-400002-1

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing, January 2015



*This book is dedicated to Jim Leu,
without whom I couldn't have written a book on Linux.*



This page intentionally left blank

Contents at a Glance

Contents ix

Preface xix

About the Author xxi

I: Packet Filtering and Basic Security Measures 1

- 1 Preliminary Concepts Underlying Packet-Filtering Firewalls 3**
- 2 Packet-Filtering Concepts 25**
- 3 iptables: The Legacy Linux Firewall Administration Program 51**
- 4 nftables: The Linux Firewall Administration Program 83**
- 5 Building and Installing a Standalone Firewall 95**

II: Advanced Issues, Multiple Firewalls, and Perimeter Networks 143

- 6 Firewall Optimization 145**
- 7 Packet Forwarding 179**
- 8 NAT—Network Address Translation 197**
- 9 Debugging the Firewall Rules 211**
- 10 Virtual Private Networks 229**

III: Beyond iptables and nftables 235

- 11 Intrusion Detection and Response 237**
- 12 Intrusion Detection Tools 249**
- 13 Network Monitoring and Attack Detection 263**
- 14 Filesystem Integrity 295**

IV: Appendices 311**A Security Resources 313****B Firewall Examples and Support Scripts 315****C Glossary 351****D GNU Free Documentation License 363****Index 371**

Contents

Preface xix

About the Author xxi

I: Packet Filtering and Basic Security Measures 1

1 Preliminary Concepts Underlying Packet-Filtering Firewalls 3

The OSI Networking Model 5

Connectionless versus Connection-Oriented Protocols 7

Next Steps 7

The Internet Protocol 7

IP Addressing and Subnetting 8

IP Fragmentation 11

Broadcasting and Multicasting 11

ICMP 12

Transport Mechanisms 14

UDP 14

TCP 14

Don't Forget Address Resolution Protocol 17

Hostnames and IP Addresses 18

IP Addresses and Ethernet Addresses 18

Routing: Getting a Packet from Here to There 19

Service Ports: The Door to the Programs on Your System 19

A Typical TCP Connection: Visiting a Remote Website 20

Summary 23

2 Packet-Filtering Concepts 25

A Packet-Filtering Firewall 26

Choosing a Default Packet-Filtering Policy 29

Rejecting versus Denying a Packet 31

Filtering Incoming Packets 31

Remote Source Address Filtering 31

Local Destination Address Filtering 34

Remote Source Port Filtering	35
Local Destination Port Filtering	35
Incoming TCP Connection State Filtering	35
Probes and Scans	36
Denial-of-Service Attacks	39
Source-Routed Packets	46
Filtering Outgoing Packets	46
Local Source Address Filtering	47
Remote Destination Address Filtering	47
Local Source Port Filtering	48
Remote Destination Port Filtering	48
Outgoing TCP Connection State Filtering	48
Private versus Public Network Services	49
Protecting Nonsecure Local Services	50
Selecting Services to Run	50
Summary	50
3 iptables: The Legacy Linux Firewall Administration Program	51
Differences between IPFW and Netfilter Firewall Mechanisms	51
IPFW Packet Traversal	52
Netfilter Packet Traversal	54
Basic iptables Syntax	54
iptables Features	55
NAT Table Features	58
mangle Table Features	60
iptables Syntax	61
filter Table Commands	62
filter Table Target Extensions	67
filter Table Match Extensions	68
nat Table Target Extensions	79
mangle Table Commands	81
Summary	82
4 nftables: The Linux Firewall Administration Program	83
Differences between iptables and nftables	83
Basic nftables Syntax	83

nftables Features	84
nftables Syntax	85
Table Syntax	85
Chain Syntax	86
Rule Syntax	87
Basic nftables Operations	91
nftables File Syntax	92
Summary	93
5 Building and Installing a Standalone Firewall	95
The Linux Firewall Administration Programs	96
Build versus Buy: The Linux Kernel	97
Source and Destination Addressing Options	98
Initializing the Firewall	99
Symbolic Constants Used in the Firewall Examples	100
Enabling Kernel-Monitoring Support	101
Removing Any Preexisting Rules	103
Resetting Default Policies and Stopping the Firewall	104
Enabling the Loopback Interface	105
Defining the Default Policy	106
Using Connection State to Bypass Rule Checking	107
Source Address Spoofing and Other Bad Addresses	108
Protecting Services on Assigned Unprivileged Ports	112
Common Local TCP Services Assigned to Unprivileged Ports	113
Common Local UDP Services Assigned to Unprivileged Ports	116
Enabling Basic, Required Internet Services	117
Allowing DNS (UDP/TCP Port 53)	118
Enabling Common TCP Services	122
Email (TCP SMTP Port 25, POP Port 110, IMAP Port 143)	123
SSH (TCP Port 22)	128
FTP (TCP Ports 21, 20)	130
Generic TCP Service	133

- Enabling Common UDP Services 134
 - Accessing Your ISP's DHCP Server (UDP Ports 67, 68) 134
 - Accessing Remote Network Time Servers (UDP Port 123) 136
- Logging Dropped Incoming Packets 138
- Logging Dropped Outgoing Packets 138
- Installing the Firewall 139
 - Tips for Debugging the Firewall Script 139
 - Starting the Firewall on Boot with Red Hat and SUSE 140
 - Starting the Firewall on Boot with Debian 141
 - Installing a Firewall with a Dynamic IP Address 141
- Summary 141

II: Advanced Issues, Multiple Firewalls, and Perimeter Networks 143

6 Firewall Optimization 145

- Rule Organization 145
 - Begin with Rules That Block Traffic on High Ports 145
 - Use the State Module for ESTABLISHED and RELATED Matches 146
 - Consider the Transport Protocol 146
 - Place Firewall Rules for Heavily Used Services as Early as Possible 147
 - Use Traffic Flow to Determine Where to Place Rules for Multiple Network Interfaces 147
- User-Defined Chains 148
- Optimized Examples 151
 - The Optimized iptables Script 151
 - Firewall Initialization 153
 - Installing the Chains 155
 - Building the User-Defined EXT-input and EXT-output Chains 157
 - tcp-state-flags 165
 - connection-tracking 166
 - local-dhcp-client-query and remote-dhcp-server-response 166

source-address-check	167
destination-address-check	168
Logging Dropped Packets with iptables	168
The Optimized nftables Script	170
Firewall Initialization	170
Building the Rules Files	172
Logging Dropped Packets with nftables	175
What Did Optimization Buy?	176
iptables Optimization	176
nftables Optimization	177
Summary	177
7 Packet Forwarding	179
The Limitations of a Standalone Firewall	179
Basic Gateway Firewall Setups	181
LAN Security Issues	182
Configuration Options for a Trusted Home LAN	183
LAN Access to the Gateway Firewall	184
LAN Access to Other LANs: Forwarding Local Traffic among Multiple LANs	186
Configuration Options for a Larger or Less Trusted LAN	188
Dividing Address Space to Create Multiple Networks	188
Selective Internal Access by Host, Address Range, or Port	190
Summary	195
8 NAT—Network Address Translation	197
The Conceptual Background of NAT	197
NAT Semantics with iptables and nftables	201
Source NAT	203
Destination NAT	205
Examples of SNAT and Private LANs	206
Masquerading LAN Traffic to the Internet	206
Applying Standard NAT to LAN Traffic to the Internet	208
Examples of DNAT, LANs, and Proxies	209
Host Forwarding	209
Summary	210

9 Debugging the Firewall Rules 211

- General Firewall Development Tips 211
- Listing the Firewall Rules 213
 - `iptables` Table Listing Example 213
 - `nftables` Table Listing Example 216
- Interpreting the System Logs 217
 - `syslog` Configuration 217
 - Firewall Log Messages: What Do They Mean? 220
- Checking for Open Ports 223
 - `netstat -a [-n -p -A inet]` 224
 - Checking a Process Bound to a Particular Port with `fuser` 226
 - Nmap 227
- Summary 227

10 Virtual Private Networks 229

- Overview of Virtual Private Networks 229
- VPN Protocols 229
 - PPTP and L2TP 229
 - IPsec 230
- Linux and VPN Products 232
 - Openswan/Libreswan 233
 - OpenVPN 233
 - PPTP 233
- VPN and Firewalls 233
- Summary 234

III: Beyond `iptables` and `nftables` 235**11 Intrusion Detection and Response 237**

- Detecting Intrusions 237
- Symptoms Suggesting That the System Might Be Compromised 238
 - System Log Indications 239
 - System Configuration Indications 239
 - Filesystem Indications 240
 - User Account Indications 240
 - Security Audit Tool Indications 241
 - System Performance Indications 241

What to Do If Your System Is Compromised	241
Incident Reporting	243
Why Report an Incident?	243
What Kinds of Incidents Might You Report?	244
To Whom Do You Report an Incident?	246
What Information Do You Supply?	246
Summary	247

12 Intrusion Detection Tools **249**

Intrusion Detection Toolkit: Network Tools	249
Switches and Hubs and Why You Care	250
ARPWatch	251
Rootkit Checkers	251
Running Chkrootkit	251
What If Chkrootkit Says the Computer Is Infected?	253
Limitations of Chkrootkit and Similar Tools	253
Using Chkrootkit Securely	254
When Should Chkrootkit Be Run?	255
Filesystem Integrity	255
Log Monitoring	256
Swatch	256
How to Not Become Compromised	257
Secure Often	257
Update Often	258
Test Often	259
Summary	261

13 Network Monitoring and Attack Detection **263**

Listening to the Ether	263
Three Valuable Tools	264
TCPDump: A Simple Overview	265
Obtaining and Installing TCPDump	266
TCPDump Options	267
TCPDump Expressions	269
Beyond the Basics with TCPDump	272

Using TCPDump to Capture Specific Protocols	272
Using TCPDump in the Real World	272
Attacks through the Eyes of TCPDump	280
Recording Traffic with TCPDump	284
Automated Intrusion Monitoring with Snort	286
Obtaining and Installing Snort	287
Configuring Snort	288
Testing Snort	289
Receiving Alerts	290
Final Thoughts on Snort	291
Monitoring with ARPWatch	291
Summary	293
14 Filesystem Integrity	295
Filesystem Integrity Defined	295
Practical Filesystem Integrity	295
Installing AIDE	296
Configuring AIDE	297
Creating an AIDE Configuration File	297
A Sample AIDE Configuration File	299
Initializing the AIDE Database	300
Scheduling AIDE to Run Automatically	301
Monitoring AIDE for Bad Things	301
Cleaning Up the AIDE Database	302
Changing the Output of the AIDE Report	303
Obtaining More Verbose Output	305
Defining Macros in AIDE	306
The Types of AIDE Checks	307
Summary	310
IV: Appendices	311
A Security Resources	313
Security Information Sources	313
Reference Papers and FAQs	314
B Firewall Examples and Support Scripts	315
iptables Firewall for a Standalone System from Chapter 5	315

`nftables` Firewall for a Standalone System
from Chapter 5 328

Optimized `iptables` Firewall from Chapter 6 332

`nftables` Firewall from Chapter 6 345

C Glossary 351

D GNU Free Documentation License 363

0. Preamble 363

1. Applicability and Definitions 363

2. Verbatim Copying 365

3. Copying in Quantity 365

4. Modifications 366

5. Combining Documents 367

6. Collections of Documents 368

7. Aggregation with Independent Works 368

8. Translation 368

9. Termination 369

10. Future Revisions of this License 369

11. Relicensing 370

Index 371

This page intentionally left blank

Preface

Welcome to the fourth edition of *Linux® Firewalls*. The book looks at what it takes to build a firewall using a computer running Linux. The material covered includes some basics of networking, IP, and security before jumping into `iptables` and `nftables`, the latest firewall software in Linux.

A reader of this book should be running a Linux computer, whether standalone or as a firewall or Internet gateway. The book shows how to build a firewall for a single client computer such as a desktop and also shows how to build a firewall behind which multiple computers can be hosted on a local network.

The final part of the book shows aspects of computer and network security beyond `iptables` and `nftables`. This includes intrusion detection, filesystem monitoring, and listening to network traffic. The book is largely Linux agnostic, meaning that just about any popular flavor of Linux will work with the material with little or no adaptation.

Acknowledgments

I'd like to thank my wife, family, and friends for their unending support. Thanks also to Robert P.J. Day and Andrew Prowant for reviewing the manuscript.

This page intentionally left blank

About the Author

Steve Suehring is a technology architect specializing in Linux and Windows systems and development. Steve has written several books and magazine articles on a wide range of technologies. During his tenure as an editor at *LinuxWorld* magazine, Steve wrote and edited articles and reviews on Linux security and advocacy including a feature story on the use of Linux in Formula One auto racing.

This page intentionally left blank

This page intentionally left blank

Packet-Filtering Concepts

What is a firewall? Over the years, the term has changed in meaning. According to RFC 2647, “Benchmarking Terminology for Firewall Performance,” a firewall is “a device or group of devices that enforces an access control policy between networks.” This definition is very broad, purposefully so in fact. A firewall can encompass many layers of the OSI model and may refer to a device that does packet filtering, performs packet inspection and filtering, implements a policy on an application at a higher layer, or does any of these and more.

A nonstateful, or stateless, firewall usually performs some packet filtering based solely on the IP layer (Layer 3) of the OSI model, though sometimes higher-layer protocols are involved in this type of firewall. An example of this type of device might include a border router that sits at the edge of a network and implements one or more access lists to prevent various types of malicious traffic from entering the network. Some might argue that this type of device isn’t a firewall at all. However, it certainly appears to fit within the RFC definition.

A border router access list might implement many different policies depending on which interface the packet was received on. It’s typical to filter certain packets at the edge of the network connecting to the Internet. These packets are discussed later in this chapter.

As opposed to a stateless firewall, a stateful firewall is one that keeps track of the packets previously seen within a given session and applies the access policy to packets based on what has already been seen for the given connection. A stateful firewall implies the basic packet-filtering capabilities of a stateless firewall as well. A stateful firewall will, for example, keep track of the stages of the TCP three-way handshake and reject packets that appear out of sequence for that handshake. Being connectionless, UDP is somewhat trickier to handle for a stateful firewall because there’s no state to speak of. However, a stateful firewall tracks recent UDP exchanges to ensure that a packet that has been received relates to a recent outgoing packet.

An *Application-level gateway* (ALG), sometimes referred to as an *Application-layer gateway*, is yet another form of firewall. Unlike the stateless firewall, which has knowledge of the Network and possibly Transport layers, an ALG primarily handles Layer 7, the Application layer of the OSI model. ALGs typically have deep knowledge of the application data

being passed and can thus look for any deviation from the normal traffic for the application in question.

An ALG will typically reside between the client and the real server and will, for all intents and purposes, mimic the behavior of the real server to the client. In effect, local traffic never leaves the LAN, and remote traffic never enters the LAN.

ALG sometimes also refers to a module, or piece of software that assists another firewall. Many firewalls come with an FTP ALG to support FTP's port mode data channel, where the client tells the server what local port to connect to so that it can open the data channel. The server initiates the incoming data channel connection (whereas, usually, the client initiates all connections). ALGs are frequently required to pass multimedia protocols through a firewall because multimedia sessions often use multiple connections initiated from both ends and generally use TCP and UDP together.

ALG is a proxy. Another form of proxy is a *circuit-level proxy*. Circuit-level proxies don't usually have application-specific knowledge, but they can enforce access and authorization policies, and they serve as termination points in what would otherwise be an end-to-end connection. SOCKS is an example of a circuit-level proxy. The proxy server acts as a termination point for both sides of the connection, but the server doesn't have any application-specific knowledge.

In each of these cases, the firewall's purpose is to enforce the access control or security policies that you define. Security policies are essentially about access control—who is and is not allowed to perform which actions on the servers and networks under your control.

Though not necessarily specific to a firewall, firewalls many times find themselves performing additional tasks, some of which might include Network Address Translation (NAT), antivirus checking, event notification, URL filtering, user authentication, and Network-layer encryption.

This book covers the ideas behind a packet-filtering firewall, both static and dynamic, or stateless and stateful. Each of the approaches mentioned controls which services can be accessed and by whom. Each approach has its strengths and advantages based on the differing information available at the various OSI reference model layers.

Chapter 1, "Preliminary Concepts Underlying Packet-Filtering Firewalls," introduced the concepts and information a firewall is based on. This chapter introduces how this information is used to implement firewall rules.

A Packet-Filtering Firewall

At its most basic level, a packet-filtering firewall consists of a list of acceptance and denial rules. These rules explicitly define which packets will and will not be allowed through the network interface. The firewall rules use the packet header fields described in Chapter 1 to decide whether to forward a packet to its destination, to silently throw away the packet, or to block the packet and return an error condition to the sending machine. These rules can be based on a wide array of factors, including the source or destination IP addresses, the source and (more commonly) destination ports, and portions of individual packets such as the TCP header flags, the types of protocol, the MAC address, and more.

MAC address filtering is not common on Internet-connected firewalls. Using MAC filtering, the firewall blocks or allows only certain MAC addresses. However, in all likelihood you see only one MAC address, the one from the router just upstream from your firewall. This means that every host on the Internet will appear to have the same MAC address as far as your firewall can see. A common error among new firewall administrators is to attempt to use MAC filtering on an Internet firewall.

Using a hybrid of the TCP/IP reference model, a packet-filtering firewall functions at the Network and Transport layers, as shown in Figure 2.1.

The overall idea is that you need to very carefully control what passes between the Internet and the machine that you have connected directly to the Internet. On the external interface to the Internet, you individually filter what's coming in from the outside and what's going out from the machine as exactly and explicitly as possible.

For a single-machine setup, it might be helpful to think of the network interface as an I/O pair. The firewall independently filters what comes in and what goes out through the interface. The input filtering and the output filtering can, and likely do, have completely different rules. Figure 2.2 depicts processing for rules in a flowchart.

This sounds pretty powerful, and it is; but it isn't a surefire security mechanism. It's only part of the story, just one layer in the multilayered approach to data security. Not all application communication protocols lend themselves to packet filtering. This type of

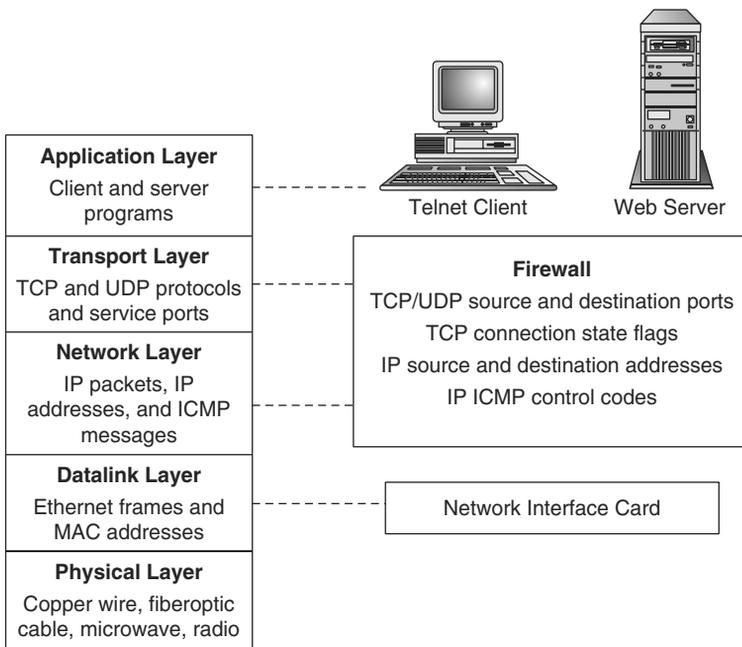


Figure 2.1 Firewall placement in the TCP/IP reference model

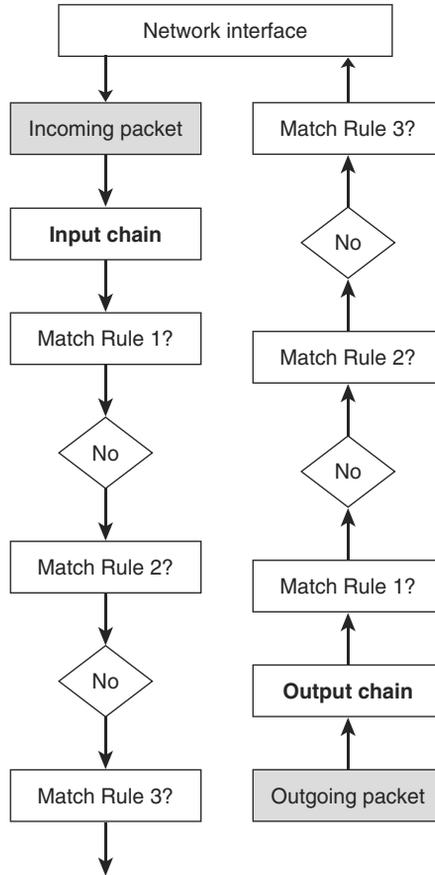


Figure 2.2 Input and output flowchart

filtering is too low-level to allow fine-grained authentication and access control. These security services must be furnished at higher levels. IP doesn't have the capability to verify that the sender is who he or she claims to be. The only identifying information available at this level is the source address in the IP packet header. The source address can be modified with little difficulty. One level up, neither the Network layer nor the Transport layer can verify that the application data is correct. Nevertheless, the packet level allows greater, simpler control over direct port access, packet contents, and correct communication protocols than can easily or conveniently be done at higher levels.

Without packet-level filtering, higher-level filtering and proxy security measures are either crippled or potentially ineffective. To some extent, at least, they must rely on the correctness of the underlying communication protocol. Each layer in the security protocol stack adds another piece that other layers can't easily provide.

Choosing a Default Packet-Filtering Policy

As stated earlier in this chapter, a firewall is a device to implement an access control policy. A large part of this policy is the decision on a default firewall policy.

There are two basic approaches to a default firewall policy:

- Deny everything by default, and explicitly allow selected packets through.
- Accept everything by default, and explicitly deny selected packets from passing through.

Without question, the deny-everything policy is the recommended approach. This approach makes it easier to set up a secure firewall, but each service and related protocol transaction that you want must be enabled explicitly (see Figure 2.3). This means that you

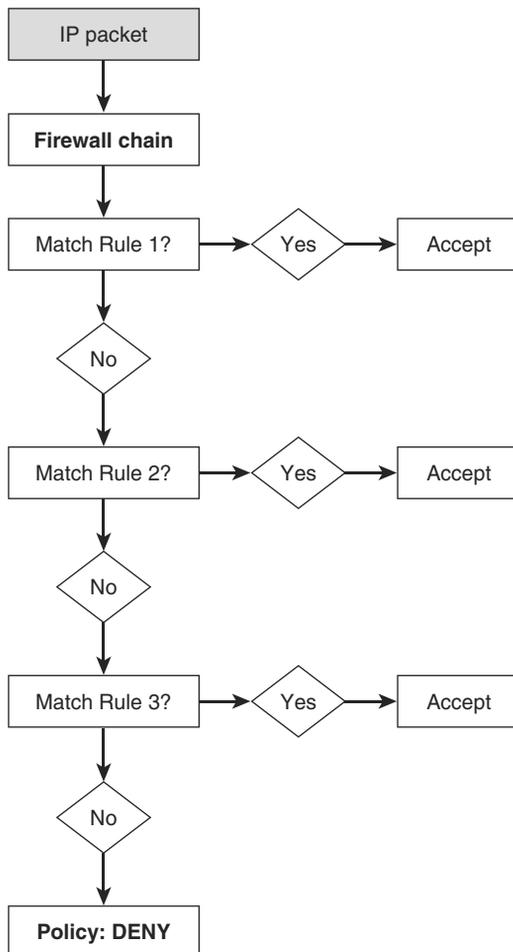


Figure 2.3 The deny-everything-by-default policy

must understand the communication protocol for each service you enable. The deny-everything approach requires more work up front to enable Internet access. Some commercial firewall products support only the deny-everything policy.

The accept-everything policy makes it much easier to get up and running right away, but it forces you to anticipate every conceivable access type that you might want to disable (see Figure 2.4). The danger is that you won't anticipate a dangerous access type until it's too late, or you'll later enable an insecure service without first blocking external access to it. In the end, developing a secure accept-everything firewall is much more work, much more difficult, almost always much less secure, and, therefore, much more error-prone.

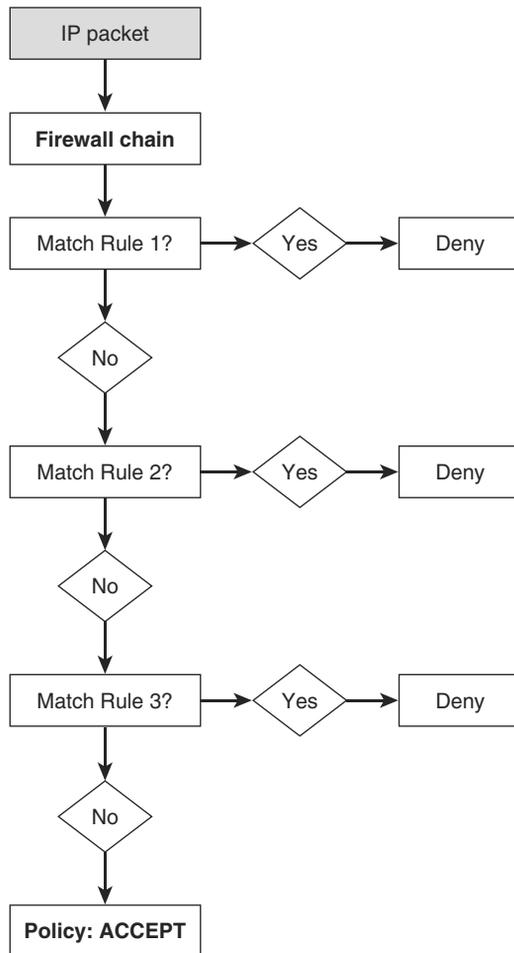


Figure 2.4 The accept-everything-by-default policy

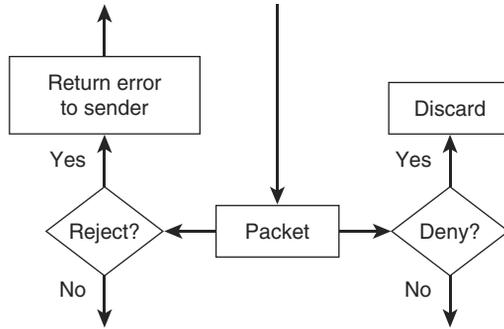


Figure 2.5 Rejecting versus denying a packet

Rejecting versus Denying a Packet

The Netfilter firewall mechanism in `iptables` and `nftables` gives you the option of either rejecting or dropping packets. What's the difference? As shown in Figure 2.5, when a packet is rejected, the packet is thrown away and an ICMP error message is returned to the sender. When a packet is dropped, the packet is simply thrown away without any notification to the sender.

Silently dropping the packet is almost always the better choice, for three reasons. First, sending an error response doubles the network traffic. The majority of dropped packets are dropped because they are malevolent, not because they represent an innocent attempt to access a service you don't happen to offer. Second, a packet that you respond to can be used in a denial-of-service (DoS) attack. Third, any response, even an error message, gives the would-be attacker potentially useful information.

Filtering Incoming Packets

The input side of the external interface I/O pair, the input rule set, is the more interesting in terms of securing your site. As mentioned earlier, you can filter based on source address, destination address, source port, destination port, TCP status flags, and other criteria. You'll learn about all these pieces of information at one point or another in the following sections.

Remote Source Address Filtering

At the packet level, the only means of identifying the IP packet's sender is the source address in the packet header. This fact allows for the possibility of source address spoofing, in which the sender places an incorrect address rather than his or her own address in the source field. The address might be a nonexistent address, or it might be a legitimate address belonging to someone else. This can allow unsavory types to break into your

system by appearing as local, trusted traffic; appearing to be you while attacking other sites; pretending to be someone else while attacking you; keeping your system bogged down responding to nonexistent addresses; or otherwise misleading you as to the source of incoming messages.

It's important to remember that you usually can't detect spoofed addresses. The address might be legitimate and routable but might not belong to the packet's sender. The next section describes the spoofed addresses you can detect.

Source Address Spoofing and Illegal Addresses

There are several major classes of source addresses you should deny on your external interface in all cases. These are incoming packets claiming to be from the following:

- **Your IP address**—You will never see legal incoming packets claiming to be from your machine. Because the source address is the only information available and it can be modified, this is one of the forms of legitimate address spoofing you can detect at the packet-filtering level. Incoming packets claiming to be from your machine are spoofed. You can't be certain whether other incoming packets are coming from where they claim to be. (Note that some operating systems crash if they receive a packet in which both the source and the destination addresses belong to the host's network interface.)
- **Your LAN addresses**—You will rarely see legal incoming packets on the external, Internet interface claiming to be *from* your LAN. It's possible to see such packets if the LAN has multiple access points to the Internet, but it would probably be a sign of a misconfigured local network. In most cases, such a packet would be part of an attempt to gain access to your site by exploiting your local trust relationships.
- **Class A, B, and C private IP addresses**—These three sets of addresses in the historical Class A, B, and C ranges are reserved for use in private LANs. They aren't intended for use on the Internet. As such, these addresses can be used by any site internally without the need to purchase registered IP addresses. Your machine should never see incoming packets from these source addresses:
 - Class A private addresses are assigned the range from 10.0.0.0 to 10.255.255.255.
 - Class B private addresses are assigned the range from 172.16.0.0 to 172.31.255.255.
 - Class C private addresses are assigned the range from 192.168.0.0 to 192.168.255.255.
- **Class D multicast IP addresses**—IP addresses in the Class D range are set aside for use as destination addresses when participating in a multicast network broadcast, such as an audiocast or a videocast. They range from 224.0.0.0 to 239.255.255.255. Your machine should never see packets from these source addresses.

- **Class E reserved IP addresses**—IP addresses in the Class E range were set aside for future and experimental use and are not assigned publicly. They range from 240.0.0.0 to 247.255.255.255. Your machine should never see packets from these source addresses—and mostly likely won't. (Because the entire address range is permanently reserved up through 255.255.255.255, the Class E range can realistically be defined as 240.0.0.0 to 255.255.255.255. In fact, some sources define the Class E address range to be exactly that.)
- **Loopback interface addresses**—The loopback interface is a private network interface used by the Linux system for local, network-based services. Rather than sending local traffic through the network interface driver, the operating system takes a shortcut through the loopback interface as a performance improvement. By definition, loopback traffic is targeted for the system generating it. It doesn't go out on the network. The loopback address range is 127.0.0.0 to 127.255.255.255. You'll usually see it referred to as 127.0.0.1, localhost, or the loopback interface, lo.
- **Malformed broadcast addresses**—Broadcast addresses are special addresses applying to all machines on a network. Address 0.0.0.0 is a special broadcast source address. A legitimate broadcast source address will be either 0.0.0.0 or a regular IP address. DHCP clients and servers will see incoming broadcast packets from source address 0.0.0.0. This is the only legal use of this source address. It is not a legitimate point-to-point, unicast source address. When seen as the source address in a regular, point-to-point, nonbroadcast packet, the address is forged, or the sender isn't fully configured.
- **Class A network 0 addresses**—As suggested previously, any source address in the 0.0.0.0 through 0.255.255.255 range is illegal as a unicast address.
- **Link local network addresses**—DHCP clients sometimes assign themselves a link local address when they can't get an address from a server. These addresses range from 169.254.0.0 to 169.254.255.255.
- **Carrier-grade NAT**—There are IPs that are marked for use by Internet providers that should never appear on a public network, the public Internet. These addresses can, however, be used in cloud scenarios, and therefore, if your server is hosted at a cloud provider, you may see these addresses. The carrier-grade NAT addresses range from 100.64.0.0 to 100.127.255.255.
- **TEST-NET addresses**—The address space from 192.0.2.0 to 192.0.2.255 is reserved for test networks.

Blocking Problem Sites

Another common, but less frequently used, source address-filtering scheme is to block all access from a selected machine or, more typically, from an entire network's IP address block. This is how the Internet community tends to deal with problem sites and ISPs that

don't police their users. If a site develops a reputation as a bad Internet neighbor, other sites tend to block it across the board.

On the individual level, blocking all access from selected networks is convenient when individuals in the remote network are habitually making a nuisance of themselves. This has historically been used as a means to fight unsolicited email, with some people going so far as to block an entire country's range of IP addresses.

Limiting Incoming Packets to Selected Remote Hosts

You might want to accept certain kinds of incoming packets from only specific external sites or individuals. In these cases, the firewall rules will define either specific IP addresses or a limited range of IP source addresses that these packets will be accepted from.

The first class of incoming packets is from remote servers responding to your requests. Although some services, such as web or FTP services, can be expected to be coming from anywhere, other services will legitimately be coming from only your ISP or specially chosen trusted hosts. Examples of servers that are probably offered only through your ISP are POP mail service, Domain Name Service (DNS) name server responses, and possible DHCP or dynamic IP address assignments.

The second class of incoming packets is from remote clients accessing services offered from your site. Again, although some incoming service connections, such as connections to your web server, can be expected to be coming from anywhere, other local services will be offered to only a few trusted remote users or friends. Examples of restricted local services might be `ssh` and `ping`.

Local Destination Address Filtering

Filtering incoming packets based on the destination address is not much of an issue. Under normal operation, your network interface card ignores regular packets that aren't addressed to it. The exception is broadcast packets, which are broadcast to all hosts on the network.

The IPv4 address `255.255.255.255` is the general broadcast destination address. It refers to all hosts on the immediate physical network segment, and it is called a *limited broadcast*. A broadcast address can be defined more explicitly as the highest address in a given subnet of IP addresses. For example, if your ISP's network address is `192.168.10.0` with a 24-bit subnet mask (`255.255.255.0`) and your IP address is `192.168.10.30`, you would see broadcast packets addressed to `192.168.10.255` from your ISP. On the other hand, if you have a smaller range of IP addresses, say a `/30` (`255.255.255.252`), then you have a total of four addresses: one network, two for hosts, and the broadcast. For example, consider the network `10.3.7.4/30`. In this network, `10.3.7.4` is the network address, the two hosts would be `10.3.7.5` and `10.3.7.6`, and the broadcast address would be `10.3.7.7`. This `/30` subnet configuration type is typically used between routers, though the actual addresses themselves may vary. The only way to know what the broadcast address will be for a given subnet is to know both an IP address within the subnet and the subnet mask. These types of broadcasts are called *directed subnet broadcasts* and are delivered to all hosts on that network.

Broadcast-to-destination address 0.0.0.0 is similar to the situation of point-to-point packets claiming to be from the broadcast source address mentioned earlier, in the section “Source Address Spoofing and Illegal Addresses.” Here, broadcast packets are directed to source address 0.0.0.0 rather than to the destination address, 255.255.255.255. In this case, there is little question about the packet’s intent. This is an attempt to identify your system as a Linux machine. For historical reasons, networking code derived from BSD UNIX returns an ICMP Type 3 error message in response to 0.0.0.0 being used as the broadcast destination address. Other operating systems silently discard the packet. As such, this is a good example of why dropping versus rejecting a packet makes a difference. In this case, the error message itself is what the probe is looking for.

Remote Source Port Filtering

Incoming requests and connections from remote clients to your local servers will have a source port in the unprivileged range. If you are hosting a web server, all incoming connections to your web server should have a source port between 1024 and 65535. (That the server port identifies the service is the intention but not the guarantee. You cannot be certain that the server you expect is running at the port you expect.)

Incoming responses and connections from remote servers that you contacted will have the source port that is assigned to the particular service. If you connect to a remote website, all incoming messages from the remote server will have the source port set to 80 (or whatever port the local client specified), the `http` service port number.

Local Destination Port Filtering

The destination port in incoming packets identifies the program or service on your computer that the packet is intended for. As with the source port, all incoming requests from remote clients to your services generally follow the same pattern, and all incoming responses from remote services to your local clients follow a different pattern.

Incoming requests and connections from remote clients to your local servers will set the destination port to the service number that you assigned to the particular service. For example, an incoming packet destined for your local web server would normally have the destination port set to 80, the `http` service port number.

Incoming responses from remote servers that you contacted will have a destination port in the unprivileged range. If you connect to a remote website, all incoming messages from the remote server will have a destination port between 1024 and 65535.

Incoming TCP Connection State Filtering

Incoming TCP packet acceptance rules can make use of the connection state flags associated with TCP connections. All TCP connections adhere to the same set of connection states. These states differ between client and server because of the three-way handshake during connection establishment. As such, the firewall can distinguish between incoming traffic from remote clients and incoming traffic from remote servers.

Incoming TCP packets from remote clients will have the `SYN` flag set in the first packet received as part of the three-way connection establishment handshake. The first connection request will have the `SYN` flag set, but not the `ACK` flag.

Incoming packets from remote servers will always be responses to the initial connection request initiated from your local client program. Every TCP packet received from a remote server will have the `ACK` flag set. Your local client firewall rules will require all incoming packets from remote servers to have the `ACK` flag set. Servers do not normally attempt to initiate connections to client programs.

Probes and Scans

A *probe* is an attempt to connect to or get a response from an individual service port. A *scan* is a series of probes to a set of different service ports. Scans are often automated.

Unfortunately, probes and scans are rarely innocent anymore. They are most likely the initial information-gathering phase, looking for interesting vulnerabilities before launching an attack. Automated scan tools are widespread, and coordinated efforts by groups of hackers are common. The security, or lack thereof, of many hosts on the Internet, along with the proliferation of worms, viruses, and zombied machines, makes scans a constant issue on the Internet.

General Port Scans

General port scans are indiscriminate probes across a large block of service ports, possibly the entire range (see Figure 2.6). These scans are somewhat less frequent—or, at least, less obvious—as more sophisticated, targeted stealth tools become available.

Targeted Port Scans

Targeted port scans look for specific vulnerabilities (see Figure 2.7). The newer, more sophisticated tools attempt to identify the hardware, operating system, and software versions. These tools are designed to identify targets that might be prone to a specific vulnerability.

Common Service Port Targets

Common targets often are individually probed as well as scanned. The attacker might be looking for a specific vulnerability, such as an insecure mail server, an unpatched web server, or an open remote procedure call (RPC) `portmap` daemon.

A more extensive list of ports can be found at <http://www.iana.org/assignments/port-numbers>. Only a few common ports are mentioned here, to give you the idea:

- Incoming packets from reserved port 0 are always bogus. This port isn't used legitimately.
- Probes of TCP ports 0 to 5 are a signature of the `sscan` program.
- `ssh` (22/tcp), `smtp` (25/tcp), `dns` (53/tcp/udp), `pop-3` (110/tcp), `imap` (143/tcp), and `snmp` (161/udp), are favorite target ports. They represent some of the most potentially vulnerable openings to a system, whether intrinsically, due to

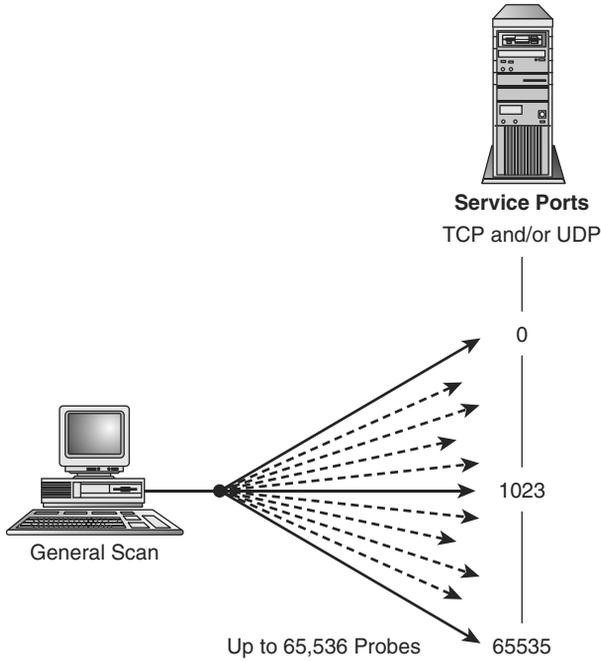


Figure 2.6 A general port scan

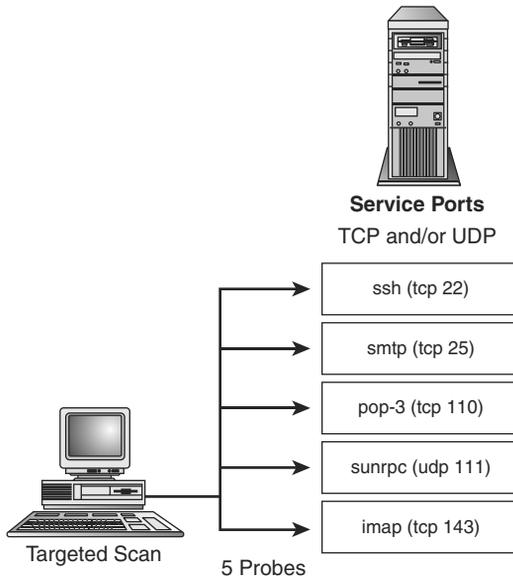


Figure 2.7 A targeted port scan

common configuration errors, or due to known flaws in the software. Because these services are so common, they are good examples of why you want to either not offer them to the outside world, or very carefully offer them with controlled outside access. NetBIOS (137-139/tcp/udp) and Server Message Block (SMB) on Windows (445/tcp) probes are tediously common. They typically pose no threat to a Linux system unless Samba is used on the system. The typical target is a Windows system, in this case, but the scans are all too common.

Stealth Scans

Stealth port scans, by definition, aren't meant to be detectable. They are based on how the TCP protocol stack responds to unexpected packets, or packets with illegal state flag combinations. For example, consider an incoming packet that has the ACK flag set but has no related connection. If the ACK were sent to a port with a listening server attached, the TCP stack wouldn't find a related connection and would return a TCP RST message to tell the sender to reset the connection. If the ACK were sent to an unused port, the system would simply return a TCP RST message as an error indication, just as the firewall might return an ICMP error message by default.

The issue is further complicated because some firewalls test only for the SYN flag or the ACK flag. If neither is set, or if the packet contains some other combination of flags, the firewall implementation might pass the packet up to the TCP code. Depending on the TCP state flag combination and the operating system receiving the packet, the system will respond with an RST or with silence. This mechanism can be used to help identify the operating system that the target system is running. In any of these cases, the receiving system isn't likely to log the event.

Inducing a target host to generate an RST packet in this manner also can be used to map a network, determining the IP addresses of systems listening on the network. This is especially helpful if the target system isn't a server and its firewall has been set to silently drop unwanted packets.

Avoiding Paranoia: Responding to Port Scans

Firewall logs normally show all kinds of failed connection attempts. Probes are the most common thing you'll see reported in your logs.

Are people probing your system this often? Yes, they are. Is your system compromised? No, it isn't. Well, not necessarily. The ports are blocked. The firewall is doing its job. These are failed connection attempts that the firewall denied.

At what point do you personally decide to report a probe? At what point is it important enough to take the time to report it? At what point do you say that enough is enough and get on with your life, or should you be writing `abuse@some.system` each time? There are no "right" answers. How you respond is a personal judgment call and depends in part on the resources available to you, how sensitive the information at your site is, and how critical the Internet connection is to your site. For obvious probes and scans, there is no clear-cut answer. It depends on your own personality and comfort level how you personally define a serious probe, and your social conscience.

With that in mind, these are some workable guidelines.

The most common attempts are a combination of automated probing, mistakes, legitimate attempts based on the history of the Internet, ignorance, curiosity, and misbehaving software.

You can almost always safely ignore individual, isolated, single connection attempts to `telnet`, `ssh`, `ftp`, `finger`, or any other port for a common service that you're not providing. Probes and scans are a fact of life on the Internet, are all too frequent, and usually don't pose a risk. They are kind of like door-to-door salespeople, commercial phone calls, wrong phone numbers, and junk postal mail. For me, at least, there isn't enough time in the day to respond to each one.

On the other hand, some probers are more persistent. You might decide to add firewall rules to block them completely, or possibly even their entire IP address space.

Scans of a subset of the ports known to be potential security holes are typically the precursor to an attack if an open port is found. More inclusive scans are usually part of a broader scan for openings throughout a domain or subnet. Current hacking tools probe a subset of these ports one after the other.

Occasionally, you'll see serious hacking attempts. This is unquestionably a time to take action. Write them. Report them. Double-check your security. Observe what they're doing. Block them. Block their IP address block.

Some system administrators take every occurrence seriously because, even if *their* machine is secure, other people's machines might not be. The next person might not even have the capability of knowing that he or she is being probed. Reporting probes is the socially responsible thing to do, for everyone's sake.

How should you respond to port scans? If you write these people, their postmaster, their uplink service provider network operations center (NOC), or the network address block coordinator, try to be polite. Give them the benefit of the doubt. Overreactions are misplaced more often than not. What might appear as a serious hacking attempt to you is often a curious kid playing with a new program. A polite word to the abuser, `root`, or postmaster can sometimes take care of the problem. More people need to be educated about Netiquette than need their network accounts rescinded. And they might be innocent of anything. Just as often, the person's system is compromised and that person has no idea what's going on and will be grateful for the information.

Probes aren't the only hostile traffic you'll see, however. Although probes are harmless in and of themselves, DoS attacks are not.

Denial-of-Service Attacks

DoS attacks are based on the idea of flooding your system with packets to disrupt or seriously degrade your Internet connection, tying up local servers to the extent that legitimate requests can't be honored or, in the worst case, crashing your system altogether. The two most common results are keeping the system too busy to do anything useful and tying up critical system resources.

You can't protect against DoS attacks completely. They can take as many different forms as the attacker's imagination allows. Anything that results in a response from your system, anything that results in your system allocating resources (including logging of the attack), anything that induces a remote site to stop communicating with you—all can be used in a DoS attack.

More on Denial-of-Service Attacks

For further information on DoS attacks, see the "Denial of Service" paper available at <http://www.cert.org>.

These attacks usually involve one of several classic patterns, including TCP SYN flooding, ping flooding, UDP flooding, fragmentation bombs, buffer overflows, and ICMP routing redirect bombs.

TCP SYN Flooding

A TCP SYN flood attack consumes your system resources until no more incoming TCP connections are possible (see Figure 2.8). The attack makes use of the basic TCP three-way handshaking protocol during connection establishment, in conjunction with IP source address spoofing.

The attacker spoofs his or her source address as a private address and initiates a connection to one of your TCP-based services. Appearing to be a client attempting to open a TCP connection, the attacker sends you an artificially generated SYN message. Your machine responds by sending an acknowledgment, a SYN-ACK. However, in this case, the

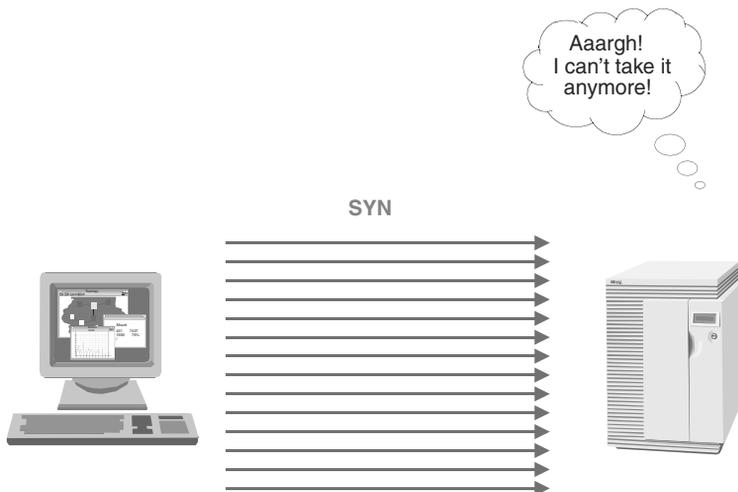


Figure 2.8 A TCP SYN flood

address that you're replying to isn't the attacker's address. In fact, because the address is private, there is no one out there to respond. The spoofed host won't return an RST message to tear down the half-opened connection.

The final stage of TCP connection establishment, receiving an ACK in response, will never happen. Consequently, finite network connection resources are consumed. The connection remains in a half-opened state until the connection attempt times out. The attacker floods your port with connection request after connection request, faster than the TCP timeouts release the resources. If this continues, all resources will be in use and no more incoming connection requests can be accepted. This applies not only to the service being probed, but to all new connections as well.

Several aids are available to Linux users. The first is source address filtering, described previously. This filters out the most commonly used spoofed source addresses, but there is no guarantee that the spoofed address falls within the categories you can anticipate and filter against.

The second is to enable your kernel's SYN cookie module, a specific retardant to the resource starvation caused by SYN flooding. When the connection queue begins to get full, the system starts responding to SYN requests with SYN cookies rather than SYN-ACKs, and it frees the queue slot. Thus, the queue never fills completely. The cookie has a short timeout; the client must respond to it within a short period before the serving host will respond with the expected SYN-ACK. The cookie is a sequence number that is generated based on the original sequence number in the SYN, the source and destination addresses and ports, and a secret value. If the response to the cookie matches the result of the hashing algorithm, the server is reasonably well assured that the SYN is valid.

Depending on the particular release, you may or may not need to enable the SYN cookie protection within the kernel by using the command `echo 1 > /proc/sys/net/ipv4/tcp_syncookies`. Some distributions and kernel versions require you to explicitly configure the option into the kernel using `make config`, `make menuconfig`, or `make xconfig` and then recompile and install the new kernel.

SYN Flooding and IP Spoofing

For more information on SYN flooding and IP spoofing, see CERT Advisory CA-96.21, "TCP SYN Flooding and IP Spoofing Attacks," at <http://www.cert.org>.

ping Flooding

Any message that elicits a response from your machine can be used to degrade your network connection by forcing the system to spend most of its time responding. The ICMP echo request message sent by ping is a common culprit. An attack called *Smurf*, and its variants, forces a system to expend its resources processing echo replies. One method of accomplishing this is to spoof the victim's source address and broadcast an echo request to an entire network of hosts. A single spoofed request message can result in hundreds or thousands of resulting replies being sent to the victim. Another way of accomplishing a similar result is to install trojans on compromised hosts across the Internet and time them

to each send echo requests to the same host simultaneously. Finally, a simple ping flood in which the attacker sends more echo requests and floods the data connection is another method for a DoS, though it's becoming less common. A typical ping flood is shown in Figure 2.9.

Ping of Death

An older exploit called the *Ping of Death* involved sending very large ping packets. Vulnerable systems could crash as a result. Linux is not vulnerable to this exploit, nor are many other current UNIX operating systems. If your firewall is protecting older systems or personal computers, those systems could be vulnerable.

The Ping of Death exploit gives an idea of how the simplest protocols and message interactions can be used by the creative hacker. Not all attacks are attempts to break into your computer. Some are merely destructive. In this case, the goal is to crash the machine. (System crashes also might be an indicator that you need to check your system for installed trojan programs. You might have been duped into loading a trojan program, but the program itself might require a system reboot to activate.)

ping is a very useful basic networking tool. You might not want to disable ping altogether. In today's Internet environment, conservative folks recommend disabling incoming ping or at least severely limiting from whom you accept echo requests. Because of ping's history of involvement in DoS attacks, many sites no longer respond to external ping requests from any but selected sources. This has always seemed to be an overreaction to the relatively small threat of a DoS based on ICMP when compared to the more ubiquitous and dangerous threats against applications and other protocols within the stack.

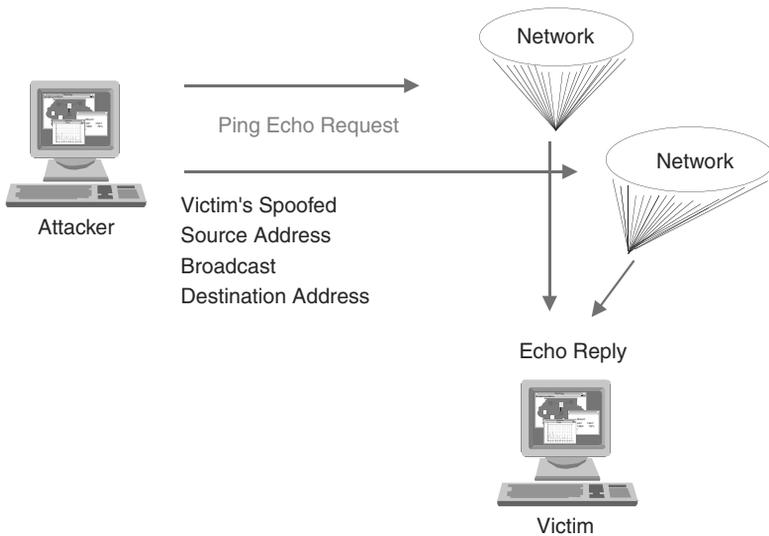


Figure 2.9 A ping flood

Dropping ping requests isn't a solution for the victim host, however. Regardless of how the recipient of the flood reacts to the packets, the system (or network) can still be overwhelmed in the process of inspecting and dropping a flood of requests.

UDP Flooding

The UDP protocol is especially useful as a DoS tool. Unlike TCP, UDP is stateless. Flow-control mechanisms aren't included. There are no connection state flags. Datagram sequence numbers aren't used. No information is maintained on which packet is expected next. There is not always a way to differentiate client traffic from server traffic based on port numbers. Without state, there is no way to distinguish an expected incoming response from an unsolicited packet arriving unexpectedly. It's relatively easy to keep a system so busy responding to incoming UDP probes that no bandwidth is left for legitimate network traffic.

Because UDP services are susceptible to these types of attacks (as opposed to connection-oriented TCP services), many sites disable all UDP ports that aren't absolutely necessary. As mentioned earlier, almost all common Internet services are TCP based. The firewall we'll build in Chapter 5, "Building and Installing a Standalone Firewall," carefully limits UDP traffic to only those remote hosts providing necessary UDP services.

The classic UDP flood attack either involves two victim machines or works in the same way the Smurf ping flood does (see Figure 2.10). A single spoofed packet from the attacker's UDP echo port, directed to a host's UDP chargen port, can result in an infinite loop of network traffic. The echo and chargen services are network test services. chargen generates an ASCII string. echo returns the data sent to the port.

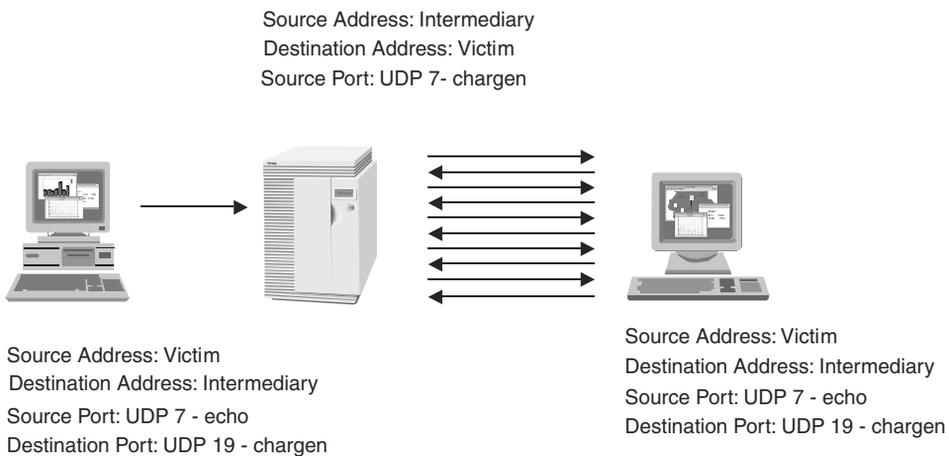


Figure 2.10 A UDP flood

UDP Port Denial-of-Service Attacks

For a fuller description of a DoS exploit using these UDP services, see CERT Advisory CA-96.01, "UDP Port Denial-of-Service Attack," at <http://www.cert.org>.

Fragmentation Bombs

Different underlying network technologies (such as Ethernet, Asynchronous Transfer Mode [ATM], and token ring) define different limits on the size of the Layer 2 frame. As a packet is passed on from one router to the next along the path from the source machine to the destination machine, network gateway routers might need to cut the packet into smaller pieces, called *fragments*, before passing them on to the next network. In a legitimate fragmentation, the first fragment contains the usual source and destination port numbers contained in the UDP or TCP transport header. The following fragments do not.

For example, although the maximum theoretical packet length is 65,535 bytes, the maximum Ethernet frame size (Maximum Transmission Unit, or MTU) is 1500 bytes.

When a packet is fragmented, intermediate routers do not reassemble the packet. The packets are reassembled either at the destination host or by its adjacent router.

Because intermediate fragmentation is ultimately more costly than sending smaller, nonfragmented packets, current systems often do MTU discovery with the target host at the beginning of a connection. This is done by sending a packet with the Don't Fragment option set in the IP header options field (the only generally legitimate current use of the IP options field). If an intermediate router must fragment the packet, it drops the packet and returns an ICMP 3 error, *fragmentation-required*.

One type of fragmentation attack involves artificially constructing very small packets. One-byte packets crash some operating systems. Current operating systems usually test for this condition.

Another use of small fragments is constructing the initial fragment so that the UDP or TCP source and destination ports are contained in the second fragment. (All networks' MTU sizes are large enough to carry a standard 40-byte IP and transport header.) Packet-filtering firewalls often allow these fragments through because the information that they filter on is not present. This form of attack is useful to get packets through the firewall that would not otherwise be allowed.

The Ping of Death exploit mentioned earlier is an example of using fragmentation to carry an illegally large ICMP message. When the ping request is reconstructed, the entire packet size is larger than 65,535 bytes, causing some systems to crash.

A classic example of a fragmentation exploit is the Teardrop attack. The method can be used to bypass a firewall or to crash a system. The first fragment is constructed to go to an allowed service. (Many firewalls don't inspect fragments after the first packet.) If it is allowed, the subsequent fragments will be passed through and reassembled by the target host. If the first packet is dropped, the subsequent packets will pass through the firewall, but the end host will have nothing to reconstruct and eventually will discard the partial packet.

The data offset fields in the subsequent fragments can be altered to overwrite the port information in the first fragment to access a disallowed service. The offset also can be altered so that offsets used in packet reassembly turn out to be negative numbers. Because kernel byte-copy routines usually use unsigned numbers, the negative value is treated as a very large positive number; the resulting copy trashes kernel memory and the system crashes.

Firewall machines and machines that do NAT for other local hosts should be configured to reassemble the packets before delivering them to the local target. Some of the `iptables` features require the system to reassemble packets before forwarding the packet to the destination host, and reassembly is done automatically.

Buffer Overflows

Buffer overflow exploits can't be protected against by a filtering firewall. The exploits fall into two main categories. The first is simply to cause a system or server to crash by overwriting its data space or runtime stack. The second requires technical expertise and knowledge of the hardware and system software or server version being attacked. The purpose of the overflow is to overwrite the program's runtime stack so that the call return stack contains a program and a jump to it. This program usually starts up a shell with `root` privilege.

Many of the current vulnerabilities in servers are a result of buffer overflows. It's important to install and keep up-to-date all the newest patches and software revisions.

ICMP Redirect Bombs

ICMP redirect message Type 5 tells the target system to change its in-memory routing tables in favor of a shorter route. Redirects are sent by routers to their adjacent hosts. Their intention is to inform the host that a shorter path is available (that is, the host and new router are on the same network, and the new router is the router that the original would route the packet to as its next hop).

Redirects arrive on an almost-daily basis. They rarely originate from the adjacent router. For residential or business sites connected to an ISP, it's very unlikely that your adjacent router will generate a redirect message.

If your host uses static routing and honors redirect messages, it's possible for someone to fool your system into thinking that a remote machine is one of your local machines or one of your ISP's machines, or even to fool your system into forwarding all traffic to some other remote host.

Denial-of-Service Attacks and Other System Resources

Network connectivity isn't the only concern in DoS attacks. Here are some examples of other areas to keep in mind while configuring your system:

- Your filesystem can overflow if your system is forced to write enormous numbers of messages to the error logs, or if your system is flooded with many copies of large email messages. You might want to configure resource limits and set up a separate partition for rapidly growing or changing filesystems.

Email Denial-of-Service Exploits

For a description of a DoS exploit using email, see “Email Bombing and Spamming” at <http://www.cert.org>.

- System memory, process table slots, CPU cycles, and other resources can be exhausted by repeated, rapid invocations of network services. You can do little about this other than setting any configurable limits for each individual service, enabling SYN cookies, and denying rather than rejecting packets sent to unsupported service ports.

Source-Routed Packets

Source-routed packets employ a rarely used IP option that allows the originator to define the route taken between two machines, rather than letting the intermediate routers determine the path. As with ICMP redirects, this feature can allow someone to fool your system into thinking that it’s talking to a local machine, an ISP machine, or some other trusted host, or to create the necessary packet flow for a man-in-the-middle attack.

Source routing has few legitimate uses in current networks. Some routers ignore the option. Some firewalls discard packets containing the option.

Filtering Outgoing Packets

If your environment represents a trusted environment, filtering outgoing packets might not appear to be as critical as filtering incoming packets. Your system won’t respond to incoming messages that the firewall doesn’t pass through. Residential sites often take this approach. Nevertheless, even for residential sites, symmetric filtering is important, particularly if the firewall protects Microsoft Windows machines. For commercial sites, outgoing filtering is inarguably important.

If your firewall protects a LAN of Microsoft Windows systems, controlling outgoing traffic becomes much more important. Compromised Windows machines have historically been (and continue to be) used in coordinated DoS attacks and other outbound attacks. For this reason especially, it’s important to filter what leaves your network.

Filtering outgoing messages also allows you to run LAN services without leaking into the Internet, where these packets don’t belong. It’s not only a question of disallowing external access to local LAN services. It’s also a question of not broadcasting local system information onto the Internet. Examples of this would be if you were running a local DHCPD, NTP, SMB, or other server for internal use. Other obnoxious services might be broadcasting `wall` or `syslogd` messages.

A related source is some of the personal computer software, which sometimes ignores the Internet service port protocols and reserved assignments. This is the personal computer equivalent of running a program designed for LAN use on an Internet-connected machine.

A final reason is simply to keep local traffic local that isn't intended to leave the LAN but that conceivably could. Keeping local traffic local is a good idea from a security standpoint but also as a means for bandwidth conservation.

Local Source Address Filtering

Filtering outgoing packets based on the source address is easy. For a small site or a single computer connected to the Internet, the source address is always your computer's IP address during normal operation. There is no reason to allow an outgoing packet to have any other source address, and the firewall should enforce this.

For people whose IP address is dynamically assigned by their ISP, a brief exception exists during address assignment. This exception is specific to DHCP and is the one case in which a host broadcasts messages using 0.0.0.0 as its source address.

For people with a LAN whose firewall machine has a dynamically assigned IP address, limiting outgoing packets to contain the source address of the firewall machine's IP address is mandatory. It protects you from several fairly common configuration mistakes that appear as cases of source address spoofing or illegal source addresses to remote hosts.

If your users or their software aren't 100% trustworthy, it's important to ensure that local traffic contains legitimate, local addresses only, to avoid participating in DoS attacks using source address spoofing.

This last point is especially important. RFC 2827, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing" (updated by RFC 3704, "Ingress Filtering for Multihomed Networks"), is a current "best practices" document speaking to exactly this point. Ideally, every router should filter out the obvious illegal source addresses and ensure that traffic leaving the local network contains only routable source addresses belonging to that network.

Remote Destination Address Filtering

As with incoming packets, you might want to allow certain kinds of outgoing packets to be addressed only to specific remote networks or individual machines. In these cases, the firewall rules will define either specific IP addresses or a limited range of IP destination addresses to which these packets will be allowed.

The first class of outgoing packets to filter by destination address is packets destined to remote servers that you've contacted. Although some packets, such as those going to web or FTP servers, can be expected to be destined to anywhere on the Internet, other remote services will legitimately be offered from only your ISP or specially chosen trusted hosts. Examples of services that are probably offered only through your ISP are mail services such as SMTP or POP3, DNS services, DHCP dynamic IP address assignment, and the Usenet news service.

The second class of outgoing packets to filter by destination address is packets destined to remote clients who are accessing a service offered from your site. Again, although some outgoing service connections, such as responses from your local web server, can be

expected to be going anywhere, other local services will be offered to only a few trusted remote sites or friends. Examples of restricted local services might be telnet, SSH, Samba-based services, and RPC services accessed via `portmap`. Not only will the firewall rules deny general incoming connections to these services, but the rules also won't allow outgoing responses from these services to just anyone.

Local Source Port Filtering

Explicitly defining which service ports on your network can be used for outgoing connections serves two purposes—one for your client programs and one for your server programs. Specifying the source ports allowed for your outgoing connections helps ensure that your programs are behaving correctly, and it protects other people from any local network traffic that doesn't belong on the Internet.

Outgoing connections from your local clients will almost always originate from an unprivileged source port. Limiting your clients to the unprivileged ports in the firewall rules helps protect other people from potential mistakes on your end by ensuring that your client programs are behaving as expected.

Outgoing packets from your local server programs will always originate from their assigned service port and will be in response to a request received. Limiting your servers to their assigned ports at the firewall level ensures that your server programs are functioning correctly at the protocol level. More important, it helps protect any private, local network services that you might be running from outside access. It also helps protect remote sites from being bothered by network traffic that should remain confined to your local systems.

Remote Destination Port Filtering

Your local client programs are designed to connect to network servers offering their services from their assigned service ports. From this perspective, limiting your local clients to connect only to their associated server's service port ensures protocol correctness. Limiting your client connections to specific destination ports serves a couple of other purposes as well. First, it helps guard against local, private network client programs inadvertently attempting to access servers on the Internet. Second, it does much to disallow outgoing mistakes, port scans, and other mischief potentially originating from your site.

Your local server programs will almost always participate in connections originating from unprivileged ports. The firewall rules limit your servers' outgoing traffic to only unprivileged destination ports.

Outgoing TCP Connection State Filtering

Outgoing TCP packet acceptance rules can make use of the connection state flags associated with TCP connections, just as the incoming rules do. All TCP connections adhere to the same set of connection states, which differs between client and server.

Outgoing TCP packets from local clients will have the `SYN` flag set in the first packet sent as part of the three-way connection establishment handshake. The initial connection request will have the `SYN` flag set, but not the `ACK` flag. Your local client firewall rules will allow outgoing packets with either the `SYN` or the `ACK` flag set.

Outgoing packets from local servers will always be responses to an initial connection request initiated from a remote client program. Every packet sent from your servers will have the `ACK` flag set. Your local server firewall rules will require all outgoing packets from your servers to have the `ACK` flag set.

Private versus Public Network Services

One of the easiest ways to inadvertently allow uninvited intrusions is to allow outside access to local services that are designed only for LAN use. Some services, if offered locally, should never cross the boundary between your LAN and the Internet beyond. Some of these services annoy your neighbors, some provide information you'd be better off keeping to yourself, and some represent glaring security holes if they're available outside your LAN.

Some of the earliest network services, the `r-*`-based commands in particular, were designed for local sharing and ease of access across multiple lab machines in a trusted environment. Some of the later services were intended for Internet access, but they were designed at a time when the Internet was basically an extended community of academicians and researchers. The Internet was a relatively open, safe place. As the Internet grew into a global network including general public access, it developed into a completely untrusted environment.

Lots of Linux network services are designed to provide local information about user accounts on the system, which programs are running and which resources are in use, system status, network status, and similar information from other machines connected over the network. Not all of these informational services represent security holes in and of themselves. It's not that someone can use them directly to gain unauthorized access to your system. It's that they provide information about your system and user accounts that can be useful to someone who is looking for known vulnerabilities. They might also supply information such as usernames, addresses, phone numbers, and so forth, which you don't want to be readily available to everyone who asks.

Some of the more dangerous network services are designed to provide LAN access to shared filesystems and devices, such as a networked printer or fax machine.

Some services are difficult to configure correctly and some are difficult to configure securely. Entire books are devoted to configuring some of the more complicated Linux services. Specific service configuration is beyond the scope of this book.

Some services just don't make sense in a home or small-office setting. Some are intended to manage large networks, provide Internet routing service, provide large database informational services, support two-way encryption and authentication, and so forth.

Protecting Nonsecure Local Services

The easiest way to protect yourself is to not offer the service. But what if you need one of these services locally? Not all services can be protected adequately at the packet-filtering level. File-sharing software, instant messaging services, and UDP-based RPC services are notoriously difficult to secure at the packet-filtering level.

One way to safeguard your computer is to not host network services on the firewall machine that you don't intend for public use. If the service isn't available, there's nothing for a remote client to connect to. Let firewalls be firewalls.

A packet-filtering firewall doesn't offer complete security. Some programs require higher-level security measures than can be provided at the packet-filtering level. Some programs are too problematic to risk running on a firewall machine, even on a less secure residential host.

Small sites such as those in the home often won't have a supply of computers available to enforce access security policies by running private services on other machines. Compromises must be made, particularly for required services that are provided solely by Linux. Nevertheless, small sites with a LAN should not be running file-sharing or other private LAN services on the firewall, such as Samba. The machine should not have unnecessary user accounts. Unneeded system software should be removed from the system. The machine should have no function other than that of a security gateway.

Selecting Services to Run

When all is said and done, only you can decide which services you need or want. The first step in securing your system is to decide which services and daemons you intend to run on the firewall machine, as well as behind the firewall in the private LAN. Each service has its own security considerations. When it comes to selecting services to run under Linux or any other operating system, the general rule of thumb is to run only network services that you need and understand. It's important to understand a network service, what it does and who it's intended for, before you run it—especially on a machine connected directly to the Internet.

Summary

Between this and the preceding chapter, the basics of networking and firewalls have been laid out. The next chapter digs deeper into `iptables` itself.

Index

: (colon), command-line syntax, 62
[] (square brackets), command-line syntax, 62
(pound sign), comment indicator, 297
| (pipe symbol), command-line syntax, 62
< > (angle brackets), command-line syntax, 62
0.0.0.0 IP address
 definition, 110
 denying packets addressed to, 109
255.255.255.255 IP address, denying packets originating from, 109

A

-a option, 224
-a (--handle) option, 85–86
-A option, 224
Accept packet and stop processing, 87
ACCEPT rule
 defining a default policy, 106
 definition, 351
accept statement, 87
accept-everything-by-default policy, 29–30, 351. *See also* Default policies.
ACK flag, 16, 351
add command
 nftables, chain syntax, 86–87
 nftables, table syntax, 85–86
Adding
 chains to a table, 86–87
 rules, 87
 tables, 85–86

Address families, 84

Address information, displaying numerically, 85–86

Address Resolution Protocol (ARP), 17–18

Addresses. See Ethernet addresses; IP addresses.

addrtype match extension, 77

AIDE (Advanced Intrusion Detection Environment)

- changing report output, 303–306
- checksum checks, 310
- configuration files, 297–300
- configuring, 297–301
- defining macros, 306–307
- grouped checks, 309
- initializing the database, 300
- installing, 296–297
- monitoring, 301–302
- report verbosity, 305–306
- running automatically, 301
- standard checks, 308–309
- types of checks, 307–310
- updating the database, 302–303

Alerts from the Snort program, 290–291

ALG (application-level gateway). See also Proxies, application-level.

- definition, 351
- description, 25–26

Angle brackets (< >), command-line syntax, 62

Application layer, 6

Applied Cryptography, 310

Arithmetic operators, 271

ARP (Address Resolution Protocol), 17–18

- arp address family, 84
- ARP header expressions, 91
- ARP packets, 18
- ARP spoofing, 264
- ARPWatch daemon, 265, 291–293

Attack detection. See also Intrusion detection.

- ARP spoofing, 264
- capturing network traffic. *See* Snort program.
- hub environment *vs.* switched, 263
- mirror ports, 263–264
- monitoring ARP traffic. *See* ARPWatch daemon.
- overview, 263–264
- packet capture and analysis. *See* TCPDump.
- span ports, 263–264

AUTH port, 351

Authentication, definition, 351

Authentication header, in VPNs, 230–231

Authorization, definition, 351

B

Basic NAT, 58, 199

Bastille Linux, 258

Bastion firewalls

- definition, 3, 354
- limitations of, 179–180
- packet forwarding, 179–180

Bidirectional NAT, 58, 199

BIND (Berkeley Internet Name Domain), 351

Bit flags, TCP, 15–16

Blocking

- directed broadcasts, 110
- limited broadcasts, 110
- local TCP services, 113–115
- problem sites, 33–34

Books and publications

- Applied Cryptography*, 310
- “Denial of Service,” 40
- “Email Bombing and Spamming,” 46
- FAQs, 314

- “Help Defeat Denial of Services Attacks: Step-by-Step,” 314
 - “Internet Firewalls: Frequently Asked Questions,” 314
 - “Multicast over TCP/IP HOW TO,” 111
 - reference papers, 314
 - RFC 1112 “Host Extensions for IP Multicasting,” 111
 - RFC 1122 “Requirements for Internet Hosts—Communication Layers,” 102
 - RFC 1458 “Requirements for Multicast Protocols,” 111
 - RFC 1631 “The IP Network Address Translator (NAT),” 197
 - RFC 1700 “Assigned Numbers,” 113
 - RFC 1812 “Requirements for IP Version 4 Routers,” 102
 - RFC 2196 “Site Security Handbook,” 238
 - RFC 2236 “Internet Group Management Protocol Version 2,” 111
 - RFC 2474, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,” 77
 - RFC 2475, “An Architecture for Differentiated Services,” 77
 - RFC 2588 “IP Multicast and Firewalls,” 111
 - RFC 2647 “Benchmarking Terminology for Firewall Performance,” 25
 - RFC 2663 “IP Network Address Translator (NAT) Terminology and Considerations,” 198
 - RFC 2827 “Network Ingress Filtering: Defeating Denial of Service Attacks . . .,” 47
 - RFC 2990 “Next Steps for the IP QoS Architecture,” 77
 - RFC 3022 “Traditional IP Network Address Translator (Traditional NAT),” 197
 - RFC 3168 “The Addition of Explicit Congestion Notification (ECN) to IP,” 77
 - RFC 3260 “New Terminology and Clarifications for Diffserv,” 77
 - RFC 3704 “Ingress Filtering for Multihomed Networks,” 47
 - security information, 313
 - “Service Name and Transport Protocol Port Number Registry (IANA),” 314
 - “Steps for Recovering from a UNIX or NT System Compromise,” 238
 - “TCP SYN Flooding and IP Spoofing Attacks,” 41
 - TCP/IP Illustrated, Volume 1, Second Edition*, 7
 - “UDP Port Denial-of-Service Attack,” 43
 - BOOTP (Bootstrap Protocol), 351**
 - BOOTPC port, 351**
 - bootpd program, definition, 351**
 - BOOTPS port, 351**
 - Border router, definition, 352**
 - Branching, 149**
 - bridge address family, 84**
 - Broadcast, definition, 352**
 - Broadcast addresses, 8, 9, 11**
 - broadcast primitive, 271**
 - Broadcasting, 11-12**
 - Buffer overflows, 45**
-
- C**
-
- CERT (Computer Emergency Response Team), 352**
 - Chain commands on individual rules, 64**
 - Chain types, nftables, 87**

Chains. See also User-defined chains.

- adding to a table, 86–87
- built-in, 61
- clearing, 85–86
- creating, 86
- definition, 352
- deleting, 86
- displaying for tables, 85–86
- FORWARD, 60–61
- INPUT, 60–61
- nat table, 61
- in nftables. *See* nftables, chain syntax.
- operations on, 62–63
- OUTPUT, 59–61
- POSTROUTING, 59–61
- PREROUTING, 59–61
- renaming, 86
- user-defined, 54–55

Checksums

- AIDE checks, 310
- definition, 352
- TCP (Transmission Control Protocol), 15

Chkrootkit program

- downloading, 251
- false negatives, 252
- false positives, 252
- infection reports, 253
- limitations, 253–254
- run schedule, 255
- running, 251–253
- using securely, 254–255

Choke firewalls, 181–182, 354**chroot program/system call, 352****CIDR (Classless Inter-Domain Routing), 352****Circuit gateway. See Proxies, circuit-level.****Class, network address, 352****Clearing chains and rules, 85****Client/server model, 352–353****CLOSED state, 17****CLOSE_WAIT state, 17****Colon (:), command-line syntax, 62****Command-line input, enabling, 85****Command-line options, nftables, 85****Commands, filter table, 62–67****Commands and subcommands, nftables, 83****Compromised machines. See Attack detection; Intrusion detection.****Computer Emergency Response Team (CERT), 352****Configuration files, AIDE, 297–300****Configuring****AIDE, 297–301**

email services. *See* Email services, initializing.

firewalls. *See* Initializing firewalls; Installing firewalls.

internal LANs, 191–192

multiple LANs, 192–195

Snort program, 288–289

Connection state, initializing firewalls, 107**Connection tracking expressions, 88–89****Connectionless vs. connection-oriented protocols, 7****connectionstate-policy file, 170, 173****connection-tracking chain, 151, 166****Conntrack expressions, 88–89****Conservation of addresses, 10****Continue processing packets, 87****continue statement, 87****Countermeasures. See Intrusion prevention.****create command, 86****CWR flag, 16**

D

Daemons. *See also specific daemons.*

- definition, 353
- listening on service ports, 19

DARPA network model, 6

Database, AIDE

- initializing, 300
- updating, 302–303

Datalink layer, 6, 353

Debian, initializing firewalls, 140

--debug option, 85

Debugging

- enabling, 85
- firewall scripts, 139–140

Debugging, firewall rules

- firewall development tips, 211–212
- fuser command, 226–227
- iptables -L INPUT, 214–215
- iptables -n -L INPUT, 215–216
- iptables table listing, 213–214
- iptables -v -L INPUT, 216
- listing firewall rules, 213–217
- log message priorities, 218
- log messages, interpreting, 220–223
- network security auditing, 227
- nftables listing example, 216–217
- Nmap tool, 227
- open ports, checking for, 223–227
- output reporting conventions, 226
- port-bound processes, checking for, 226–227
- syslog configuration, 217–220
- system logs, 217–223
- verbosity, 216

Default policies. *See also accept-everything-by-default policy; Deny-everything-by-default policy.*

- accept-everything-by-default, 29–30

- definition, 353

- deny-everything-by-default, 29–30
- packet-filtering, 31–32

delete command, nftables

- chain syntax, 86
- rule syntax, 87
- table syntax, 85

Deleting

- chains, 86
- rules, 87
- tables, 85

Demilitarized zone (DMZ), 180, 353

Demultiplexing, 6

“Denial of Service,” 40

Denial of service (DoS) attacks. *See DoS (denial of service) attacks.*

Deny-everything-by-default policy. *See also Default policies.*

- debugging, 211–212
- definition, 353
- description, 30
- shortcomings, 114

Denying packets vs. rejecting, 31

Destination addresses, NAT, 202

Destination NAT (DNAT), 205–206, 209–210

destination-address-check chain, 151, 168

Detecting intrusions. *See Intrusion detection.*

DHCP (Dynamic Host Configuration Protocol), 353

DHCPACK messages, 135

DHCPDECLINE messages, 135

DHCPDISCOVER messages, 135

DHCPINFORM messages, 135

DHCPNAK messages, 135

DHCPOFFER messages, 135

DHCPRELEASE messages, 135

DHCPREQUEST messages, 135

Directed broadcasts, 12, 110

Direction qualifier, TCPDump, 270–271
Directories, including in a search path, 85
Disallowing incoming packets, 108–109
DMZ (demilitarized zone), 180, 353
DNAT (destination NAT), 205–206, 209–210
DNAT target extensions, 56, 58, 80–81
DNS (Domain Name Service)
 definition, 18, 353
 DNS lookups, as a client, 120–121
 DNS lookups, as a forwarding server, 121–122
 enabling, 117–122
 zone transfers, 118
DNS BIND port usage, 121
DNS lookups, 120–122
DNS protocol, 119
DNS traffic, identifying, 157
dns-policy file, 170
Documentation. See Books and publications.
Domains, 18
DoS (denial of service) attacks. See also Filtering incoming packets.
 buffer overflows, 45
 countermeasures, 41
 definition, 353
 e-mail exploits, 46
 enabling the SYN cookie module, 41
 filesystem overflow, 45
 fragmentation bombs, 44–45
 ping, disabling, 42
 ping flooding, 41–42
 Ping of Death, 42–43
 redirect bombs, 45
 Smurf attack, 41
 source address filtering, 40–41
 TCP SYN flooding, 40–41

 Teardrop attack, 44
 UDP flooding, 43
Dotted decimal notation, 8
Dotted quad notation, 8
Drop packet and stop processing, 88
DROP rule
 defining a default policy, 106
 definition, 353
drop statement, 88
Dropping packets, 108, 112, 138
Dual-homed computer, 353. See also Multihomed computer.
Dynamic Host Configuration Protocol (DHCP), 353
Dynamically assigned address, 353

E

ECE flag, 16
“Email Bombing and Spamming,” 46
Email services, DoS (denial of service) attacks, 46
Email services, initializing
 hosting a mail server, 127–128
 mail protocols, 124
 overview, 123
 receiving mail, 125–127
 relaying outgoing mail, 124
 sending mail to external mail servers, 125
 sending over SMTP, 123
Encapsulation, 6
End-to-end transparency, 4
Error messages
 ICMP Type 3 error message, 35, 44
 iptables, selecting, 57
 output stream, 304
 STDERR, standard error stream, 304

ESP (encapsulating security payload), 231–232

ESTABLISHED packets, 73

ESTABLISHED state, 17

established state expression, 89

Ethernet addresses, 18

Ethernet cards, identifying, 18

Ethernet frame, 353

Exploits and attacks. *See* Attack detection; Intrusion detection.

Expressions, nftables

- ARP header expressions, 91
- IPv4 payload expressions, 90
- IPv6 header expressions, 90
- TCP header expressions, 90
- UDP header expressions, 91

Expressions, TCPDump, 269–271

Extensions. *See* Statements.

External rules files, 170

EXT-icmp-in chain, 152, 164

EXT-icmp-out chain, 152, 164

EXT-input chains, 151, 157

EXT-log-in chain, 152, 168–170

EXT-log-out chain, 152, 168–170

EXT-output chains, 151, 157

F

-f (--file) option, 85

Facilities, 217–218

Fall, Kevin R., 7

False negatives/positives, 252

File syntax, nftables, 92

File Transfer Protocol (FTP). *See* FTP (File Transfer Protocol).

Files, including, 85

Filesystem integrity

- basic integrity checks, 295
- checksums, 295–296
- definition, 295–296

- intrusion indications, 240
- software for checking, 255–256. *See also* AIDE (Advanced Intrusion Detection Environment).

Filesystem overflow, DoS (denial of service) attacks, 46

Filter, firewall rule, 354

filter chains, 87

filter table

- description, 54–55
- feature extensions, 56
- flushing the chain, 103
- match extensions, 56
- syntax. *See* iptables syntax, filter table.
- target extensions, 56

Filtering incoming packets. *See also* DoS (denial of service) attacks; Packet-filtering firewalls.

- blocking problem sites, 33–34
- illegal addresses, 32–33
- limited broadcast, 34
- limiting incoming packets to selected hosts, 34
- by local destination address, 34–35
- by local destination port, 35
- probes, 36–39
- by remote source address, 31–34
- by remote source port, 35
- scans, 36–39
- source address spoofing, 32–33
- source-routed packets, 46
- by TCP connection state, 35–36

Filtering iptables log messages, 57

Filtering outgoing packets. *See also* Packet-filtering firewalls.

- by local source address, 47
- by local source port, 48
- by outgoing TCP connection state, 48–49

Filtering outgoing packets. *See also* Packet-filtering firewalls. (*continued*)

overview, 46–47

by remote destination address, 47–48

by remote destination port, 48

FIN flag, 16

finger program, 354

FIN_WAIT_2 state, 17

Firewall Administration Program. *See* Iptables.

Firewall initialization, optimization example, 153–154, 170–172

Firewall logs. *See* Logging.

Firewall rules, listing, 213–217

Firewalls

basic. *See* Bastion firewalls.

bastion. *See* Bastion firewalls.

choke, 181–182, 354

combining with VPNs, 233–234

definition, 3, 25, 354

development tips, 211–212

dual-homed, 354

initializing. *See* Initializing firewalls.

installing. *See* Installing firewalls.

NAT-enabled routers as, 4

nonstateful. *See* Stateless firewalls.

packet-filtering. *See* Packet-filtering firewalls.

purpose of, 3–4

router devices as, 4

screened-host, 354

screened-subnet, 354

standalone. *See* Bastion firewalls.

stateful, 25

stateless, 25

transparency, 4

Firewalls, examples (code listings)

iptables, 315–328

nftables, 328–332

Flooding

packet, 354

ping, 41–42

TCP SYN, 40–41

UDP, 43

flush command, nftables

chain syntax, 86

table syntax, 85

Flushing the chains

definition, 103–104

effect on default policy, 211

nftables, chain syntax, 86

nftables, table syntax, 85

Forward, definition, 354

FORWARD chains, mangle table, 61

forward hooks, 85

Forwarding. *See also* Packet forwarding.

host, 209–210

NAT, 201

port, 59

Fragment, definition, 355

Fragmentation bombs, 44–45

Frames, OSI (Open System Interconnection), 6

FTP (File Transfer Protocol)

anonymous, 355

authenticated, 355

definition, 355

initializing firewalls, 130–133

on unprivileged ports, 114

Full NAT, 201

fuser command, 226–227

G

Gateway, definition, 355

Gateway firewall setups, packet forwarding, 181–182

gateway primitive, 271

Generic routing encapsulation, 230

goto statement, 88

greater primitive, 271

Grouped checks, AIDE, 309

H

-h (--help) option, 85

Hacks. *See* Attack detection; Intrusion detection.

--handle (-a) option, 85–86

header expressions

 ARP, 91

 IPv6, 90

 TCP, 90

 UDP, 91

Header flags, TCP, 283

Headers

 authentication, VPNs, 230–231

 IPv4 addressing, 8

 IPv6, 8

 TCP, 15

Help, displaying, 85

Host forwarding, example, 209–210

Hostnames, IP addressing, 18

hosts.allow, TCP wrappers' configuration file, 355

hosts.deny, TCP wrappers' configuration file, 355

HOWTO documents, 355

hping3 program, 260

HTTP (Hypertext Transfer Protocol), 355

HTTP conversations, capturing, 273–277

Hub environment vs. switched, 263

Hubs

 definition, 355

 intrusion detection, 250

I

-i (--interactive) option, 85

-I (--includepath) option, 85

IANA (Internet Assigned Numbers Authority), 19–20, 355

ICMP (Internet Control Message Protocol), 12–14, 355

icmp filter table match options, 66–67

ICMP traffic, optimization example, 163–165

ICMP Type 3 error message, 35, 44

icmp-policy file, 170

identd server, 355

IGMP (Internet Group Management Protocol), 111

IKE (Internet Key Exchange), 232

Illegal addresses, 32–33

IMAP (Internet Message Access Protocol), 355

Impossible addresses, logging, 102

Incident reporting, intrusions detected.
 See Intrusion response, incident reporting.

Including files, 85

Incoming multicast packets, 111

Incoming packets. *See* Filtering incoming packets.

inet address family, 84

inetd server, definition, 355

Infection reports, 253

Initializing firewalls. *See also* Installing firewalls.

 connection state, 107

 on Debian, 140

 DNS (Domain Name Service),
 enabling, 117–122

 flushing the chain, 103–104

 FTP, 130–133

 generic TCP service, 133–134

 impossible addresses, logging, 102

Initializing firewalls. See also Installing firewalls. *(continued)*

- inadvertent lockout, 100
- Internet services, enabling, 117–122
- kernel-monitoring, enabling, 101–102
- logging, 108, 109
- log_martians command, 102
- loopback interface, enabling, 105
- preexisting rules, removing from chains. *See* Flushing the chain.
- on Red Hat, 140
- redirect messages, disabling, 102
- remotely, 100
- rule checking, bypassing, 107
- rule invocations, 99–100
- scalability, 107
- source address validation, disabling, 102
- source-routed packets, disabling, 101
- spoofing source addresses, 108–112
- SSH (Secure Shell), 128–130
- stopping the firewall, 104–105
 - on SUSE, 140
- SYN cookies, enabling, 102
- TCP services, enabling, 122–128
- timeouts, 107
- UDP services, enabling, 134–138

Initializing firewalls, bad addresses

- address 0.0.0.0, 109–110
- address 255.255.255.255, 109
- directed broadcasts, 110
- disallowing incoming packets, 108–109
- dropping packets, 108, 112, 138
- incoming multicast packets, 111
- limited broadcasts, 110
- logging dropped packets, 138
- multicast packets with non-UDP protocol, 111

- multicast registration and routing, 111–112
- spoofed multicast network packets, 110–111

Initializing firewalls, default policies

- defining, 106
- resetting, 104–105
- rules, 106

Initializing firewalls, email services

- hosting a mail server, 127–128
- mail protocols, 124
- overview, 123
- receiving mail, 125–127
- relaying outgoing mail, 124
- sending mail to external mail servers, 125
- sending over SMTP, 123

Initializing firewalls, shell script

- executing, 99–100
- symbolic constants for names and addresses, 100

INPUT chains, mangle table, 61**input hooks, 84****insert command, 87****Installing**

- AIDE, 296–297
- Snort program, 287–288
- TCPDump, 266–267
- user-defined chains, optimization example, 155–156

Installing firewalls. See also Initializing firewalls.

- with dynamic IP addresses, 141
- firewall script, 139–140
- start argument, 139
- starting and stopping the firewall, 140–141
- stop argument, 139

- Internet Assigned Numbers Authority (IANA), 19–20, 355
- Internet Control Message Protocol (ICMP), 12–14, 355
- Internet Group Management Protocol (IGMP), 111
- Internet Key Exchange (IKE), 232
- Internet Message Access Protocol (IMAP), 355
- Internet Protocol (IP), 7, 12–14
- Internet Protocol Security (IPSec), 230. *See also* IP addresses.
- Internet services, enabling, 117–122
- Intrusion detection. *See also* Attack detection; Intrusion response.
 - human role in, 237–238
 - overview, 237–238
- Intrusion detection, symptoms
 - filesystem indications, 240
 - overview, 238–239
 - security audit tool indications, 241
 - system configuration indications, 239–240
 - system log indications, 239
 - system performance indications, 241
 - user account indications, 240–241
- Intrusion detection toolkit
 - establishing traffic baselines, 250
 - filesystem integrity software, 255–256
 - hubs, 250
 - limitations of tools, 253–254
 - log monitoring, 256–257
 - monitoring SSH login failures, 256–257
 - network sniffers, 249
 - network tools, 249–250
 - ntop program, 250
 - rootkit checkers, 251
 - rootkits, 251
 - Snort, 249–250
 - Swatch program, 256–257
 - switches, 250
 - TCPDump, 249
- Intrusion detection toolkit, Chkrootkit program
 - downloading, 251
 - false negatives, 252
 - false positives, 252
 - infection reports, 253
 - limitations, 253–254
 - run schedule, 255
 - running, 251–253
 - using securely, 254–255
- Intrusion prevention
 - Bastille Linux, 258
 - DoS (denial of service) attacks, 41
 - hping3 program, 260
 - Nikto program, 260
 - Nmap (Network Mapper) program, 259–260
 - open ports, testing for, 260
 - overview, 257
 - penetration testing, 259–260
 - secure often, 257–258
 - test often, 259–260
 - update often, 258–259
 - web servers, testing, 260
- Intrusion response
 - checklist, 242–243
 - documenting your actions, 242
 - keeping a log, 242
 - overview, 241–243
 - snapshot the system logs, 242
- Intrusion response, incident reporting
 - designating a report recipient, 246

Intrusion response, incident reporting*(continued)*

- kinds of reportable incidents, 244–245
- overview, 243
- reasons for, 243–244
- recommended information, 246–247

INVALID packets, 73**invalid state expression, 89****invalid-policy file, 170, 173****IP (Internet Protocol), 7, 12–14****ip address family, 84****IP addresses. *See also* IPsec.**

- 0.0.0.0, 109–110
- 255.255.255.255, 109
- bad addresses. *See* Initializing firewalls, bad addresses.
- broadcast, 8, 9, 11
- broadcasting, 11–12
- conservation of addresses, 10
- directed broadcasts, 12
- DNS (Domain Name Service), 18
- domains, 18
- dotted decimal notation, 8
- dotted quad notation, 8
- Ethernet addresses, 18
- hostnames, 18
- illegal, 32–33
- for IPv4, 8–9
- for IPv6, 8
- limited broadcasts, 8, 12
- linking physical devices to IP addresses. *See* ARP (Address Resolution Protocol).
- loopback, 8
- masking, 99
- multicast, 9–10
- multicasting, 11–12
- network, 8

- network domains, 18
- network-directed broadcasts, 8
- overview, 8–11
- special, 7
- subnetting, 8–11
- subscribers, 11–12
- symbolic names for, 18, 98
- syntax, 9
- unicast, 9

IP datagrams

- definition, 355
- maximum size, 11
- MTU (Maximum Transmission Unit), 11
- splitting. *See* IP fragmentation.

IP fragmentation, 11**ip6 address family, 84****ipchains module**

- definition, 355
- in earlier distributions, 96
- vs.* iptables, 52

IPFW (IP firewall) mechanism. *See also***ipfwadm module.**

- definition, 355
- vs.* Netfilter, 51–54. *See also* Netfilter firewall.
- in older distributions, 96
- packet traversal, 52–54

ipfwadm module, 96, 356**iprange match extension, 77–78****IPsec (Internet Protocol Security), 230. *See also* IP addresses.****\$IPT, 97****iptables**

- definition, 356
- error messages, selecting, 57
- filter log messages, 57
- filter table, 54–56

- firewall example (code listing), 315–328
- vs.* ipchains, 52
- L INPUT, 214–215
- load feature, potential bugs, 140–141
- mangle table, 54–55, 56, 60–61
- match extensions, 56
- n -L INPUT, 215–216
- NAPT (Network Address and Port Translation), 58
- vs.* nftables, 83
- packet matching, 57
- QUEUE target, 57
- REJECT target, 57
- RETURN target, 58
- save feature, potential bugs, 140–141
- script, optimization example, 151–152
- shell script, shebang line (first line), 97
- syntax, 54–55
- TOS field, 57
- user-defined chains, 54–55

iptables command

- defining rules, 97
- definition, 54
- enabling filter table commands, 62
- location, setting, 97
- syntax, 55

iptables syntax

- chain commands on individual rules, 64
- FORWARD chains, 61
- icmp filter table match options, 66–67
- INPUT chains, 61
- list chain command, options, 63
- LOG target extension, 67
- mangle table, 61, 81–82
- mark target extension, 81–82
- nat table, 61

- nat table target extensions, 79–82
- OUTPUT chains, 61
- POSTROUTING chains, 61
- PREROUTING chains, 61
- primary tables, 61
- REJECT target extension, 68

iptables syntax, filter table

- addrtype match extension, 77
- built-in chains, 61
- commands, 62–67
- iprange match extension, 77–78
- length match extension, 78–79
- limit match extension, 70–71
- LOG target extension, 67
- mac match extension, 75
- mark match extension, 76
- match extensions, 68–79
- match operations, 62, 64
- multiport match extension, 69–70
- operations on entire chains, 62–63
- operations on rules, 62
- owner match extension, 75–76
- REJECT target extension, 68
- rule options, 64–65
- state match extension, 71–75
- target extensions, 67–68
- tcp match options, 65
- tos match extension, 76–77
- udp match options, 66
- ULOG target extension, 57, 68
- unclean match extension, 77

iptables syntax, nat table

- DNAT target extensions, 80–81
- MASQUERADE target extensions, 80
- REDIRECT target extensions, 81
- SNAT target extensions, 79–80
- target extensions, 79–81

iptables table listing, 213–214

iptables -v -L INPUT, 216

IPv4 addressing

- address shortage, 4
- classes, 8–9
- dotted decimal notation, 8
- dotted quad notation, 8
- header, 8
- IP addressing, 8–9

IPv4 payload expressions, 90

IPv6

- header, 8
- header expressions, 90
- IP addressing, 8

J

-j LOG target, 108, 138

jump statement, 88

K

Kernel-monitoring, enabling, 101–102

klogd daemon, 356

L

-L command, 223

L2TP (Layer 2 Tunneling Protocol), 229–230

LAND attack, 284

LANs (local area networks)

- definition, 356
- NAT example, 209–210
- security, packet forwarding, 182–183

LANs, packet forwarding on a larger or less trusted

- configuring an internal LAN, 191–192
- configuring multiple LANs, 192–195
- creating multiple networks, 188–190

- dividing address space, 188–190

- overview, 188

- selective internal access, 190–195

- subnetting, 188–190

LANs, packet forwarding on a trusted

- forwarding local traffic, 186–188

- LAN access to the gateway firewall, 184–186

- multiple LANs, 186–188

length match extension, 78–79

less primitive, 271

Libreswan program, 233

limit match extension, 70–71

Limit reached on matching received packets, 88

limit statement, 88

Limited broadcasts

- blocking, 110
- broadcasting, 12
- definition, 8
- filtering incoming packets, 34

Limiting incoming packets to selected hosts, 34

Linux

- output streams, 304
- VPNs (Virtual Private Networks), 232–233

Linux Firewall Administration Program. See iptables.

Linux kernel, custom vs. stock, 97–98

list chain command, options, 63

list command, nftables

- chain syntax, 86
- table syntax, 85–86

Local destination address, filtering by, 34–35

Local destination port, filtering incoming packets, 35

Local source address, filtering outgoing packets, 47

Local source port, filtering outgoing packets, 48

local_dhcp_client_query chain, 166–167

local_dns_client_request chain, 159–161

local_dns_server_query chain, 151, 158

localhost, definition, 356

localhost-policy file, 170, 172

local_tcp_client_request chain, 152

local_tcp_server_response chain, 152, 161–162

local_udp_client_request chain, 152, 163

lockd daemon, 116

Lockout, inadvertent, 100, 139–140

Log messages

- filtering, 57
- interpreting, 220–223
- priorities, 218

Log monitoring for intrusion detection, 256–257

Log packets, 88

log statement, 88

LOG target extension, 67

Logging (administrator’s journal), 242

Logging (system logs)

- as debugging tools, 217–223
- dropped packets, 138, 168–170, 175–176
- initializing firewalls, 108, 109
- intrusion indications, 239
- port scans, 38
- snapshotting as intrusion response, 242

log_martians command, 102

log-policy file, 170

log-tcp-state chain, 152, 165–166

Loopback addresses, 8

Loopback interface, 105, 356

M

MAC addresses

- definition, 17
- identifying Ethernet cards, 18
- packet-filtering firewalls, 27

mac match extension, 75

Macros, in AIDE, 306–307

Man pages, definition, 356

mangle table

- built-in chains, 61
- command syntax, 81–82
- description, 54–55
- FORWARD chains, 60
- INPUT chains, 60
- MARK extension, 56, 81–82
- marking, 60
- OUTPUT chains, 60
- POSTROUTING chains, 60
- PREROUTING chains, 60
- target extensions, 56
- TOS extension, 56
- TOS field, 60

MARK extension, 56

mark match extension, 76

mark target extension, 81–82

Masking IP addresses, 99

MASQUERADE, 203–204

MASQUERADE target extensions, 57, 59, 80

Masquerading. *See also* NAPT (Network Address and Port Translation); NAT (Network Address Translation).

- definition, 356
- description, 201
- in earlier Linux versions, 52
- in iptables, 59
- LAN traffic to the Internet, example, 206–208

Match extensions, filter table, 68–79
Match operations, filter table, 62, 64
MD5 algorithm, 356
Meta expressions, 89
Mirror ports, 263–264
Monitoring
 AIDE, 301–302
 ARP traffic. *See* ARPWatch daemon.
 for automated intrusion detection. *See*
 Snort program.
 kernel-monitoring, enabling, 101–102
 logs, 256–257
 networks with ARPWatch daemon,
 291–293
 sessions, 72
 Snort alerts, 290–291
 SSH login failures, 256–257
 system logs, 256–257
MSS (Maximum Segment Size), 17
MTU (Maximum Transmission Unit), 11, 356
Multicast addresses, 9–10
“Multicast over TCP/IP HOW TO,” 111
Multicast packets, 111, 356
Multicast registration and routing, 111–112
Multicasting, 11–12
**Multihomed computer, 356. *See also* Dual-
 homed computer.**
multiport match extension, 69–70, 114

N

-n option, 224
-n (--numeric) option, 85–86
Name server, primary, 356
Name server, secondary, 356
Naming firewall scripts, 139
**NAPT (Network Address and Port
 Translation), 58, 199. *See also*
 Masquerading.**

**NAT (Network Address Translation). *See also*
 Masquerading.**
 advantages of, 199–200
 basic, 199
 bidirectional, 199
 definition, 4, 356–357
 destination addresses, 202
 disadvantages of, 200
 DNAT (destination NAT), 205–206
 forwarding, 201
 full, 201
 introduction, 197–198
 in iptables, 52
 MASQUERADE, 203–204
 masquerading, 201
 NAPT (Network Address and
 Port Translation), 199. *See also*
 Masquerading.
 nat table syntax, 203
 REDIRECT destination NAT,
 205–206
 semantics, 201–206
 SNAT (source NAT), 203–204
 source NAT. *See* SNAT (source
 NAT).
 traditional, 198–199
 with transport-mode IPSec, 200
 twice, 199
NAT (Network Address Translation), examples
 DNAT (destination NAT), 209–210
 host forwarding, 209–210
 LANs, 209–210
 masquerading LAN traffic to the
 Internet, 206–208
 proxies, 209–210
 SNAT and private LANs, 206–208
 standard NAT, LAN traffic on the
 Internet, 208

nat chains, 87**nat table**

- basic NAT, 58
- bidirectional NAT, 58
- built-in chains, 61
- definition, 54–55
- DNAT target extensions, 56, 58
- feature overview, 58–60
- flushing the chain, 103
- MASQUERADE target extensions, 57, 59
- NAPT (Network Address and Port Translation), 58
- OUTPUT chains, 59
- port forwarding, 59
- POSTROUTING chains, 59
- PREROUTING chains, 59
- REDIRECT target extensions, 57
- RETURN target, 58
- SNAT target extensions, 56, 58, 59
- syntax, 203. *See also* iptables syntax, nat table.
- target extensions, 56, 56–58, 79–82
- traditional unidirectional outbound NAT, 58
- twice NAT, 58

NAT-enabled routers as firewalls, 4**Netfilter firewall mechanism**

- definition, 357
- as firewall administration program, 96
- vs.* IPFW, 51–54
- packet traversal, 54

Netfilter Tables. *See* Nftables.**netstat command, 224–226****netstat program, 357****Network Address and Port Translation (NAPT), 58, 199. *See also* Masquerading.****Network address class, 352****Network Address Translation (NAT). *See* NAT (Network Address Translation).****Network addresses, 8****Network domains, 18****Network File System (NFS), 357****Network layer, 6, 357****Network Mapper (Nmap)**

- definition, 357
- description, 227
- identifying open ports and available devices, 259–260, 281–282
- intrusion detection, 259–260

Network models, 6. *See also* OSI (Open System Interconnection) model layers.**Network News Transfer Protocol (NNTP), 357****Network security auditing, 227****Network sniffers, 249****Network Time Protocol (NTP), 137, 357****Network tools, 249–250****Network-directed broadcasts, 8****Networks. *See also* LANs; VPNs (Virtual Private Networks).**

- private *vs.* public, 50
- protecting nonsecure local services, 50
- selecting services to run, 50
- subnetting, 8–11

NEW packets, 73**new state expression, 89****NFS (Network File System), 357****nft command syntax, 83****nft program**

- definition, 357
- as firewall administration program, 96
- version number, displaying, 85

nftables

- add command, 86–87
- adding chains to a table, 86–87
- address families, 84

nftables (*continued*)

- arp address family, 84
- bridge address family, 84
- chain syntax, 86–87
- chain types, 87
- clearing chains, 86
- command-line options, 85
- create command, 86
- creating chains, 86
- definition, 357
- delete command, 86
- deleting chains, 86
- displaying rules in a chain, 86
- file syntax, 92
- as firewall administration program, 96
- firewall example (code listing), 328–332
- flush command, 86
- forward hooks, 85
- inet address family, 84
- input hooks, 84
- ip address family, 84
- ip6 address family, 84
- vs.* iptables, 83
- list command, 86
- listing, example, 216–217
- nft command syntax, 83
- output hooks, 85
- postrouting hooks, 85
- prerouting hooks, 84
- rename command, 86
- renaming chains, 86
- rule subcommand, 84
- script, optimization example, 170
- table subcommand, 84
- typical commands and subcommands, 83

nftables, basic operations

- ARP header expressions, 91
- IPv4 payload expressions, 90
- IPv6 header expressions, 90
- TCP header expressions, 90
- UDP header expressions, 91

nftables, command-line options

- a (--handle) option, 85–86
- address information, displaying numerically, 85–86
- debug option, 85
- debugging, enabling, 85
- directories, including in a search path, 85
- enabling command-line input, 85
- f (--file) option, 85
- h (--help) option, 85
- help, displaying, 85
- i (--interactive) option, 85
- I (--includepath) option, 85
- including files, 85
- n (--numeric) option, 85–86
- nft version number, displaying, 85
- port information, displaying numerically, 85–86
- rule handles, displaying, 85–86
- v (--version) option, 85

nftables, expressions

- connection tracking expressions, 88–89
- established state expression, 89
- invalid state expression, 89
- meta expressions, 89
- new state expression, 89
- payload expressions, 88
- related state expression, 89
- state expressions, 89
- untracked state expression, 89

nftables, rule syntax

- accept packet and stop processing, 87
- accept statement, 87
- add command, 87
- adding rules, 87
- continue processing packets, 87
- continue statement, 87
- delete command, 87
- deleting rules, 87
- drop packet and stop processing, 88
- drop statement, 88
- goto statement, 88
- insert command, 87
- jump statement, 88
- limit reached on matching received packets, 88
- limit statement, 88
- log packets, 88
- log statement, 88
- prepend a rule on a chain, 87
- queue statement, 88
- reject statement, 88
- return processing to the calling chain, 88
- return statement, 88
- send processing to a specified chain, don't return, 88
- send processing to a specified chain, return, 88
- statements and verdicts, 87–88
- stop and reject the packet, 88
- stop and send packets to the user-space process, 88

nftables, table syntax

- add command, 85–86
- adding a table, 85–86
- clearing all chains and rules for a table, 85

- default tables, 85–86
- delete command, 85
- deleting a table, 85
- displaying chains and rules for a table, 85–86
- flush command, 85
- list command, 85–86
- table syntax, 85–86

nft-vars file, 170**Nikto program, 260****Nmap (Network Mapper)**

- definition, 357
- description, 227
- identifying open ports and available devices, 259–260, 281–282
- intrusion detection, 259–260

NNTP (Network News Transfer Protocol), 357**NS flag, 16****ntop program, 250****NTP (Network Time Protocol), 137, 357****ntpd daemon, 137****--numeric (-n) option, 85–86**

O
Open Shortest Path First (OSPF), 357**Open System Interconnection (OSI) model layers. See OSI (Open System Interconnection) model layers.****Openswan program, 233****OpenVPN program, 233****Optimization**

- external rules files, 170
- goal of, 176–177
- ICMP traffic, 163–165
- iptables, 176–177
- nftables, 177
- rc.firewall script, 173–175

Optimization (*continued*)

- source address checking, bypassing, 162–163
- TCP, enabling local server traffic, 161
- TCP traffic, enabling from local clients, 159–161
- UDP, local client traffic, 162

Optimization, examples

- connection-tracking chain, 166
- destination-address-check chain, 168
- EXT-icmp-in chain, 164
- EXT-icmp-out chain, 164
- EXT-input chains, building, 157
- EXT-log-in chain, 168–170
- EXT-log-out chain, 168–170
- EXT-output chains, building, 157
- firewall initialization, 153–154, 170–172
- ICMP traffic, 163–165
- installing user-defined chains, 155–156
- iptables firewall (code listing), 332–345
- iptables script, 151–152
- local_dhcp_client_query chain, 166–167
- local_dns_client_request chain, 159–161
- local_dns_server_query chain, 158
- local_tcp_server_response chain, 161–162
- local_udp_client_request chain, 163
- logging dropped packets, 168–170
- log-tcp-state chain, 165–166
- nftables firewall (code listing), 345–349
- nftables script, 170
- rc.firewall script, 345–349
- remote_dhcp_server_response chain, 166–167
- remote_dns_server_query chain, 158

- remote_dns_server_response chain, 159–161
- remote_tcp_client_request chain, 161–162
- remote_udp_server_response chain, 163
- source address checking, bypassing, 162–163
- source-address-check chain, 167–168
- TCP, enabling local server traffic, 161
- TCP traffic, enabling from local clients, 159–161
- tcp-state-flags chain, 165
- UDP, local client traffic, 162

Optimization, rule organization

- building rules files, 172–176
- bypassing spoofing rules, 146
- connection state, enabling, 173
- creating tables, 172
- external rules files, 170
- heavily used services, 147
- ICMP rules, placing, 147
- ICMP traffic, 175
- incoming packet rules, placing, 146–147
- invalid traffic, dropping, 173
- local client traffic, over TCP, 174
- local server traffic, over TCP, 175
- localhost traffic, enabling, 172
- rc.firewall script, 173–175
- state module for ESTABLISHED and RELATED matches, 146
- traffic, enabling, 173–174
- traffic flow to determine rule placement, 147–148
- transport protocols, 146–147
- UDP rules, placing, 147
- where to begin, 145

Optimization, user-defined chains

- branching, 149
- characteristics of, 150–151
- connection-tracking, 151, 166
- destination-address-check, 151, 168
- DNS traffic, identifying, 157
- EXT-icmp-in, 152, 164
- EXT-icmp-out, 152, 164
- EXT-input, 151
- EXT-log-in, 152, 168–170
- EXT-log-out, 152, 168–170
- EXT-output, 151
- local_dhcp_client_query, 166–167
- local_dns_client_request, 159–161
- local_dns_server_query, 151, 158
- local_tcp_client_request, 152
- local_tcp_server_response, 152, 161–162
- local_udp_client_request, 152, 163
- logging dropped packets, 168–170, 175–176
- log-tcp-state, 152, 165–166
- remote_dhcp_server_response, 152, 166–167
- remote_dns_server_query, 158
- remote_dns_server_response, 151, 159–161
- remote_tcp_client_request, 152, 161–162
- remote_tcp_server_response, 152
- remote_udp_server_response, 152, 163
- source-address-check, 151, 167–168
- tcp-state-flags, 151, 165
- USER_CHAINS variable, 151

OSI (Open System Interconnection) model layers

- Application, 6
- connectionless *vs.* connection-oriented protocols, 7

- Datalink, 6
- definition, 357
- demultiplexing, 6
- encapsulation, 6
- frames, 6
- Network, 6
- overview, 5–6
- Physical, 6
- Presentation, 6
- Session, 6
- Transport layer, 6
- Transport protocols. *See* TCP (Transmission Control Protocol); UDP (User Datagram Protocol).

OSPF (Open Shortest Path First), 357**Outgoing TCP connection state, filtering outgoing packets, 48–49****OUTPUT chains**

- mangle table, 61
- nat table, 59, 61

output hooks, 85**owner match extension, 75–76**

P
-p option, 224**Packet forwarding**

- choke firewalls, 181–182
- DMZ (demilitarized zone), 180
- gateway firewall setups, 181–182
- LAN security, 182–183
- limitations of a bastion firewall, 179–180
- perimeter networks, 180

Packet forwarding, on a larger or less trusted LAN

- configuring an internal LAN, 191–192
- configuring multiple LANs, 192–195
- creating multiple networks, 188–190

Packet forwarding, on a larger or less trusted LAN (*continued*)

- dividing address space, 188–190
- overview, 188
- selective internal access, 190–195
- subnetting, 188–190

Packet forwarding, on a trusted home LAN

- forwarding local traffic, 186–188
- LAN access to the gateway firewall, 184–186
- multiple LANs, 186–188

Packet matching, iptables, 57**Packet-filtering firewalls. See also Filtering incoming packets; Filtering outgoing packets.**

- accept-everything-by-default policy, 29–30
- default policy, 29–30
- deny-everything-by-default policy, 29–30
- MAC address filtering, 27
- overview, 26–28
- rejecting packets *vs.* denying packets, 31

Packets

- definition, 357
- destination address, specifying, 98–99
- dropped, logging, 138
- dropping, 108, 112, 138
- filtering. *See* Filtering incoming packets; Filtering outgoing packets; Packet-filtering firewalls.
- fragments, 44
- logging, 88
- source address, specifying, 98–99

PATH variable, 357**Payload expressions, 88****Peer-to-peer communication protocol, 357****Penetration testing, 259–260****Penetrations. See Attack detection; Intrusion detection.****Perimeter networks, packet forwarding, 180****Physical devices, linking to IP addresses. See ARP (Address Resolution Protocol).****Physical layer, 6, 357****PID (process ID), 358****ping flooding, 41–42****Ping of Death, 42–43****pings**

- capturing, 279
- definition, 358
- disabling, 42

Pipe symbol (|), command-line syntax, 62**Point-to-Point Tunneling Protocol (PPTP), 229–230, 233****Policy defaults. See Default policies.****POP (Post Office Protocol), 358****Port forwarding, 59****Port information, displaying numerically, 85–86****Port numbers**

- mapping to service names, 19–20
- numeric *vs.* symbolic, 96–97

Port scans

- definition, 36–38, 358
- in firewall logs, 38
- general, 36
- for open ports, 281–282
- responding to, 38–39
- stealth, 38
- targeted, 36–38
- threat level, 38–39

portmap daemon

- definition, 358
- description, 113

Ports

- common scan targets, 36–38
- definition, 358

privileged, 358. *See also* Unprivileged ports.

service ports, 19–23

Ports, open

checking for, 223–227

scanning for, 281–282

testing for, 260

Post Office Protocol (POP), 358

POSTROUTING chains, 59, 61

postrouting hooks, 85

Pound sign (#), comment indicator, 297

PPTP (Point-to-Point Tunneling Protocol), 229–230, 233

pptpd daemon, 233

Prepend a rule on a chain, 87

PREROUTING chains, 59, 61

prerouting hooks, 84

Presentation layer, 6

Primitives for TCPDump, 271

Private networks vs. public, 49–50

Probes, definition, 36, 358

Process ID (PID), 358

Protocol qualifier, TCPDump, 271

Proxies

application-level, 358. *See also* ALG (application-level gateway).

circuit-level, 358–359

definition, 358

example, 209–210

PSH flag, 16

Public networks vs. private, 49–50

Q

QoS (Quality of Service), 359

Queries, capturing, 279

queue statement, 88

QUEUE target, 57

R

RARP (Reverse Address Resolution Protocol), 359

rc.firewall script, 173–175

Recording traffic, 284–286

Red Hat, initializing firewalls, 140

Redirect bombs, 45

REDIRECT destination NAT, 205–206

redirect messages, disabling, 102

REDIRECT target extensions, 57, 81

REJECT rule, 106, 359

reject statement, 88

REJECT target extension, 57, 68

Rejecting packets vs. denying, 31

RELATED packets, 73

related state expression, 89

Remote destination address, filtering outgoing packets, 47–48

Remote destination port, filtering outgoing packets, 48

Remote procedure call (RPC), 359

Remote source address, filtering incoming packets, 31–34

Remote source port, filtering incoming packets, 35

remote_dhcp_server_response chain, 152, 166–167

remote_dns_server_query chain, 158

remote_dns_server_response chain, 151, 159–161

Remotely initializing firewalls, 100

remote_tcp_client_request chain, 152, 161–162

remote_tcp_server_response chain, 152

remote_udp_server_response chain, 152, 163

rename command, 86

Renaming chains, 86

- Report output, AIDE, 303–306
- Report verbosity, AIDE, 305–306
- Reporting intrusions. See Intrusion response, incident reporting.
- Request For Comments (RFC), 359
- Resolver, 359
- Resources. See Books and publications.
- Responding to intrusions. See Intrusion response.
- Return processing to the calling chain, 88
- return statement, 88
- RETURN target, 58
- Reverse Address Resolution Protocol (RARP), 359
- RFC (Request For Comments), 359
- RFC 1112 “Host Extensions for IP Multicasting,” 111
- RFC 1122 “Requirements for Internet Hosts—Communication Layers,” 102
- RFC 1458 “Requirements for Multicast Protocols,” 111
- RFC 1631 “The IP Network Address Translator (NAT),” 197
- RFC 1700 “Assigned Numbers,” 113
- RFC 1812 “Requirements for IP Version 4 Routers,” 102
- RFC 2196 “Site Security Handbook,” 238
- RFC 2236 “Internet Group Management Protocol Version 2,” 111
- RFC 2474, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,” 77
- RFC 2475, “An Architecture for Differentiated Services,” 77
- RFC 2588 “IP Multicast and Firewalls,” 111
- RFC 2647 “Benchmarking Terminology for Firewall Performance,” 25
- RFC 2663 “IP Network Address Translator (NAT) Terminology and Considerations,” 198
- RFC 2827 “Network Ingress Filtering: Defeating Denial of Service Attacks . . .,” 47
- RFC 2990 “Next Steps for the IP QoS Architecture,” 77
- RFC 3022 “Traditional IP Network Address Translator (Traditional NAT),” 197
- RFC 3168 “The Addition of Explicit Congestion Notification (ECN) to IP,” 77
- RFC 3260 “New Terminology and Clarifications for Diffserv,” 77
- RFC 3704 “Ingress Filtering for Multihomed Networks,” 47
- RIP (Routing Information Protocol), 359
- Rootkit checkers, 251
- Rootkits, 251
- route chains, 87
- Router devices as firewalls, 4
- Routing protocols, 19
- RPC (remote procedure call), 359
- RST flag, 16
- Rule organization
 - building rules files, 172–176
 - bypassing spoofing rules, 146
 - connection state, enabling, 173
 - creating tables, 172
 - external rules files, 170
 - heavily used services, 147
 - ICMP rules, placing, 147
 - ICMP traffic, 175
 - incoming packet rules, placing, 146–147
 - invalid traffic, dropping, 173
 - local client traffic, over TCP, 174
 - local server traffic, over TCP, 175
 - localhost traffic, enabling, 172
 - rc.firewall script, 173–175
 - state module for ESTABLISHED and RELATED matches, 146
 - traffic, enabling, 173–174

traffic flow to determine rule placement, 147–148
 transport protocols, 146–147
 UDP rules, placing, 147
 where to begin, 145

rule subcommand, 84

Rules. See also Filter, firewall; Firewall; nftables, rule syntax.

adding, 87
 checking, bypassing, 107
 clearing, 85
 definition order, 96
 deleting, 87
 filter table operations on, 62
 flushing the chain, 103
 handles, displaying, 85–86
 invocations, initializing firewalls, 99–100
 listing, 85–86, 213–217
 options, filter table, 64–65
 packet addresses, specifying, 98–99
 prepending on a chain, 87
 removing from chains. *See* Flushing the chain.

Runlevel, 359

S

Scalability, initializing firewalls, 107

Scanning for open ports. See Port scans.

Schneier, Bruce, 310

Screened host. See Firewalls, screened-host.

Screened subnet. See Firewalls, screened-subnet.

Scripts, definition, 359

Secure Shell (SSH) protocol. See SSH (Secure Shell) protocol.

Secure Sockets Layer (SSL) protocol, 360

Securing often, as intrusion prevention, 257–258

Security associations, VPNs, 232

Security audit tools, intrusion indications, 241

Segments, TCP, 15, 359

Send processing to a specified chain, don't return, 88

Send processing to a specified chain, return, 88

Server programs. See Daemons.

Service names, mapping to port numbers, 19–20

Service ports, 19–23

Session layer, 6

Session monitoring, maintaining state information, 72

setgid program, 359

setuid program, 359

setup-tables file, 170, 172

Shebang line (first script line), 97

Shell, definition, 359

SMTP (Simple Mail Transfer Protocol), 360

SMTP conversations, capturing, 277–278

Smurf attack, 41, 282–283

Snapshotting the system logs, 242

SNAT (source NAT)

and private LANs, example, 206–208

semantics, 203–204

target extensions, nat table, 56, 58, 59, 79–80

SNMP (Simple Network Management Protocol), 360

Snort program

configuring, 288–289

description, 249–250, 265

installing, 287–288

Snort program, automated intrusion monitoring

obtaining, 287–288

overview, 286

- Snort program, automated intrusion monitoring** (*continued*)
 - receiving alerts, 290–291
 - with Swatch, 290–291
 - testing, 289–290
- Socket, definition, 360**
- SOCKS package, 360**
- Source address checking, bypassing, 162–163**
- Source addresses**
 - filtering, 41
 - spoofing. *See* Spoofing source addresses.
 - validation, disabling, 102
- source-address-check chain, 151, 167–168**
- Source-routed packets**
 - disabling, 101
 - filtering, 46
- Span ports, 263–264**
- Special addresses, 7**
- Splitting IP datagrams. *See* IP fragmentation.**
- Spoofing source addresses**
 - definition, 360
 - initializing firewalls, 108–112
 - multicast network packets, 110–111
 - overview, 32–33
- Square brackets ([]), command-line syntax, 62**
- squid Web cache, blocking TCP-based services, 114**
- SSH conversations, capturing, 278**
- SSH (Secure Shell) protocol**
 - definition, 360
 - initializing firewalls, 128–130
 - login failures, monitoring, 256–257
- SSL (Secure Sockets Layer) protocol, 360**
- Standalone firewall. *See* Bastion firewall.**
- Standard checks, AIDE, 308–309**
- Standard NAT, LAN traffic on the Internet, 208**
- start argument, 139**
- Starting and stopping firewalls, 140–141**
- State expressions, 89**
- state match extension, 71–75**
- Stateless firewalls, 25**
- Statements, rule syntax, 87–88**
- Statements and verdicts, 87–88**
- Statistically assigned address, 360**
- STDERR, standard error stream, 304**
- STDIN, standard input stream, 304**
- STDOUT, standard output stream, 304**
- Stealth port scans, 38**
- “Steps for Recovering from a UNIX or NT System Compromise,” 238**
- Stevens, Richard W., 7**
- stop argument, 139**
- Stop processing rules and**
 - reject the packet, 88
 - send packets to the user-space process, 88
- Stopping and starting firewalls, 104–105, 140–141**
- Subnet layer, definition, 360**
- Subnetting, 10–11**
- Subscribers, 11–12**
- SUSE, initializing firewalls, 140**
- Swatch program, 256–257, 290–291**
- Switched environment vs. hub, 263**
- Switches, intrusion detection, 250**
- Symbolic names for**
 - hosts, 98
 - IP addresses, 18, 98
 - port numbers, 96–97
- SYN cookies, enabling, 41, 102**
- SYN flag, 16, 360**
- SYN packets, 16**

SYN segments, 16
SYN_ACK packets, 17
SYN_ACK segments, 17
SYN_RCVD state, 17
SYN_SENT state, 16
syslog configuration, 217–220
syslog.conf file, 360
syslogd daemon, 360
System configuration, intrusion indication, 239–240
System logs
 as debugging tools, 217–223
 dropped packets, 138, 168–170, 175–176
 initializing firewalls, 108, 109
 intrusion indications, 239
 port scans, 38
 snapshotting as intrusion response, 242
System performance, intrusion indications, 241

T

table subcommand, 84
Table syntax, nftables, 85–86
Tables
 adding, 85–86
 clearing chains and rules, 85
 deleting, 85
 displaying chains and rules, 85–86
Target extensions
 filter table, 67–68
 mangle table, 56
 nat table, 56, 56–58, 79–81
Targeted port scans, 36–38
TCP (Transmission Control Protocol)
 a typical connection, 20–23
 bit flags, 15–16. *See also specific flags.*

 checksums, 15
 CLOSED state, 17
 CLOSE_WAIT state, 17
 connections, 16–17
 definition, 14–15, 360
 enabling, 122–128
 ESTABLISHED state, 17
 FIN_WAIT_2 state, 17
 generic, initializing firewalls, 133–134
 header, 15
 MSS (Maximum Segment Size), 17
 protocol tables, 116–117
 segments, 15
 SYN packets, 16
 SYN segments, 16
 SYN_ACK packets, 17
 SYN_ACK segments, 17
 SYN_RCVD state, 17
 SYN_SENT state, 16
 three-way handshake, 16
 TIME_WAIT state, 17
 traffic, enabling from local clients, 159–161
TCP, enabling local server traffic, 161
TCP connection state, filtering incoming packets, 35–36
TCP header expressions, 90
TCP header flags, 283
tcp match options, 65
TCP SYN flooding, 40–41
“TCP SYN Flooding and IP Spoofing Attacks,” 41
tcp-client-policy file, 170
TCPDump
 arithmetic operators, 271
 broadcast primitive, 271
 description, 249, 265
 direction qualifier, 270–271

TCPDump (*continued*)

- expressions, 269–271
- gateway primitive, 271
- greater primitive, 271
- installing, 266–267
- less primitive, 271
- obtaining, 266–267
- options, 266–269
- overview, 265–266
- primitives, 271
- protocol qualifier, 271
- type qualifier, 269–270

TCPDump, attack detection

- LAND attack, 284
- Nmap (Network Mapper), 281–282
- overview, 280–281
- recording traffic, 284–286
- scanning for open ports, 281–282
- Smurf attacks, 282–283
- TCP header flags, 283
- Xmas Tree attack, 283

TCPDump, capturing

- HTTP conversations, 273–277
- other TCP-based protocols, 278–279
- pings, 279
- queries, 279
- SMTP conversations, 277–278
- SSH conversations, 278

TCP/IP reference model

- definition, 360
- firewall placement, 27
- layers, 6

tcp-server-policy file, 170**tcp-state-flags chain, 151, 165****tcp_wrapper scheme, 360****Teardrop attack, 44****Testing**

- as intrusion prevention, 259–260
- for open ports, 260
- penetration, 259–260
- Snort program, 289–290
- web servers, 260

TFTP (Trivial File Transfer Protocol), 360**Three-way handshake, 16, 361****Time To Live (TTL), 361****Timeouts, initializing firewalls, 107****TIME_WAIT state, 17****TOS (Type of Service), 361****TOS extension, 56****TOS field, 57****tos match extension, 76–77****traceroute tool, 361****Traditional NAT, 198–199****Traditional unidirectional outbound NAT, 58****Traffic baselines, establishing, 250****Transmission Control Protocol (TCP).
See TCP (Transmission Control Protocol).****Transparency, firewalls, 4****Transport layer, 6****Transport layer, definition, 361****Transport mode, VPNs, 231****Transport protocols. See TCP (Transmission Control Protocol); UDP (User Datagram Protocol).****Trivial File Transfer Protocol (TFTP), 360****TTL (Time To Live), 361****Tunnel mode, VPNs, 231****Tuples, 72****Twice NAT, 58, 199****Type of Service (TOS), 361****Type qualifier, 269–270**

U

UDP (User Datagram Protocol)

- definition, 14, 361
- flooding, 43
- local client traffic, 162
- protocol tables, 116–117

UDP (User Datagram Protocol), enabling

- accessing your ISP's DHCP server, 134–136
- DHCP message types, 135
- DHCP protocol, 136
- overview, 134–138
- remote network time servers, accessing, 136–138

UDP header expressions, 91

udp match options, 66

"UDP Port Denial-of-Service Attack," 43

ULOG target extension, 57, 68

unclean match extension, 77

Unicast, definition, 361

Unicast addresses, 9

Unprivileged ports

- definition, 358
- official port number assignments, 113
- port range, syntax, 114
- port scans, 113
- purpose of, 19

Unprivileged ports, protecting services on

- blocking local TCP services, 113–115
- common local TCP services, 113–115
- common local UDP services, 116
- deny-by-default, shortcomings, 114
- disallowing connections, 114–115
- FTP, 114
- official port number assignments, 113
- overview, 112–113
- port range, syntax, 114

port scans, 113

TCP service protocol tables, 116–117

UDP service protocol tables, 116–117

untracked state expression, 89

Updating, as intrusion prevention, 258–259

URG flag, 16

User accounts, intrusion indications, 240–241

User Datagram Protocol (UDP). See UDP (User Datagram Protocol).

USER_CHAINS variable chain, 151

User-defined chains

- branching, 149
- characteristics of, 150–151
- connection-tracking, 151, 166
- destination-address-check, 151, 168
- DNS traffic, identifying, 157
- EXT-icmp-in, 152, 164
- EXT-icmp-out, 152, 164
- EXT-input, 151
- EXT-log-in, 152, 168–170
- EXT-log-out, 152, 168–170
- EXT-output, 151
- local_dhcp_client_query, 166–167
- local_dns_client_request, 159–161
- local_dns_server_query, 151, 158
- local_tcp_client_request, 152
- local_tcp_server_response, 152, 161–162
- local_udp_client_request, 152, 163
- logging dropped packets, 168–170, 175–176
- log-tcp-state, 152, 165–166
- remote_dhcp_server_response, 152, 166–167
- remote_dns_server_query, 158
- remote_dns_server_response, 151, 159–161
- remote_tcp_client_request, 152, 161–162

User-defined chains (*continued*)

- remote_tcp_server_response, 152
- remote_udp_server_response, 152, 163
- source-address-check, 151, 167–168
- tcp-state-flags, 151, 165
- USER_CHAINS variable, 151

UUCP protocol, 361

V

-v (--version) option, 85

Verbosity of output, 216

VPNs (Virtual Private Networks)

- authentication header, 230–231
- combining with firewalls, 233–234
- ESP (encapsulating security payload), 231–232
- generic routing encapsulation, 230
- IKE (Internet Key Exchange), 232
- IPSec (Internet Protocol Security), 230
- L2TP (Layer 2 Tunneling Protocol), 229–230
- Libreswan program, 233
- Linux, 232–233

- Openswan program, 233
- OpenVPN program, 233
- overview, 229
- PPTP (Point-to-Point Tunneling Protocol), 229–230, 233
- protocols, 229–232
- security associations, 232
- transport mode, 231
- tunnel mode, 231

W

Web servers, testing, 260

World-readable, 361

World-writable, 361

X

X Windows, 361

Xmas Tree attack, 283

Z

Zone transfers, 118