

 WILEY

Intrusion Detection

Network
Security
Beyond the
Firewall

Terry Escamilla



PAN EX1037

Palo Alto Networks v. BT Americas

IPR2023-00889

Intrusion Detection

Network Security Beyond the Firewall

Terry Escamilla

Wiley Computer Publishing



John Wiley & Sons, Inc.

New York ♦ Chichester ♦ Weinheim ♦ Brisbane ♦ Singapore ♦ Toronto

Publisher: Robert Ipsen
Editor: Carol Long
Assistant Editor: Pam Sobotka
Managing Editor: Brian Snapp
Electronic Products, Associate Editor: Mike Sosa
Text Design & Composition: D&G Limited, LLC

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This book is printed on acid-free paper. ☺

Copyright © 1998 by Terry Escamilla. All rights reserved.

Published by John Wiley & Sons, Inc.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

Library of Congress Cataloging-in-Publication Data:

Escamilla, Terry, 1956—

Instruction detection : network security beyond the firewall / Terry Escamilla.

p. cm.

Includes index.

ISBN 0-471-29000-9 (alk. paper)

1. Computer networks--Security measures. 2. Computer security.

I. Title.

TK5105.59.E83 1998

C05.8--dc21

98-33703

CIP

Printed in the United States of America.

10 9 8 7 6 5 4 3 2

Contents

Preface	x
Acknowledgments	xiii
Introduction	xv
Overview of the Book and Intrusion Detection	xv
Who Should Read This Book	xv
How the Book Is Organized	xvi
The Reality of Tradeoffs	xix
Part 1 Before Intrusion Detection: Traditional Computer Security	1
1 Intrusion Detection and the Classic Security Model	3
Back to Basics: The Classic Security Model	3
Goals of Computer Security	4
Learn to Ask Tough Questions	6
A Basic Computer Security Model	9
The Reference Monitor	10
What Makes a Good Reference Monitor	12
Enhancing the Security Model Further	14
Identification and Authentication (I&A)	14
Access Control	18
Auditing	20
Classifying Security Products with a Nod to Intrusion Detection	21
Identification and Authentication	22
Access Control	22
Scanners	22
Intrusion Detection and Monitoring	23
Additional Product Differences	23
Prevention, Detection, and Response with Intrusion Detection	25
Where to Go from Here	26

2	The Role of Identification and Authentication in Your Environment	29
	Identification and Authentication in UNIX	30
	Users and Groups	30
	Superuser	32
	What Are the Subjects in UNIX?	33
	UNIX Login	34
	UNIX Password Mechanism	35
	Storing Passwords in a Central Server	37
	Identification and Authentication in NT	39
	Users and Groups in NT	39
	Subjects in NT	40
	NT Login Security	40
	NT Authentication Using a Domain Controller	41
	How Hackers Exploit Weaknesses in Password Security	42
	Easily Guessed Passwords	44
	Brute Force Attacks	44
	Social Engineering	47
	Trojan Horses	49
	Network Sniffing	49
	Electromagnetic Emissions Monitoring	50
	Software Bugs	51
	Improving upon I&A with Authentication Servers	52
	Third-Party Authentication	52
	<i>A Cryptography Primer</i>	53
	Ideas for Improving I&A Security	71
	One-Time Passwords	72
	Strong Authentication	72
	<i>One-Time Passwords and One-Time Pads</i>	73
	Two-Factor Authentication	74
	Challenge-Response Authentication	77
	The Need for Intrusion Detection	78
	<i>Biometrics</i>	78
3	The Role of Access Control in Your Environment	81
	Configuration Problems	82
	Program Bugs	82
	What Is Access Control?	84
	How Are Access Control Decisions Made?	85
	Access Control Lists	86

Who Are You?	87
Access Control in UNIX	87
Who Are You in the UNIX Environment?	87
UNIX File and Directory Permissions	89
<i>Are You Remembering to Ask Tough Questions?</i>	90
<i>Link Counts, Hard Links, and Symbolic Links</i>	92
Increasing Your Privileges or Capabilities	94
Background Processes and Credentials	96
Access Control in NT	97
NT Rights and Privileges	97
<i>Who Are You in NT?</i>	97
Permissions for NT Files and Directories	98
How Hackers Get around Access Control	102
How to Improve upon Access Control	104
Memco SeOS	105
APIs	107
Impact of SeOS on Base Operating System Security	107
SeOS Auditing	108
Other SeOS Features	109
Going beyond SeOS	110
Why You Still Need Intrusion Detection	111
4 Traditional Network Security Approaches	113
Layers of Network Security	114
Security between Layers on a System	117
Security between Peer Layers across Systems	117
I&A for Network Security Entities	119
<i>How Hackers Exploit Protocols</i>	119
How Many Network Entities Are There?	120
I&A for Users and Groups in a Network	122
<i>Security Models within Models</i>	122
Network Node I&A	124
Software Can Be a Network Entity	125
Network Access Control	126
Network Application Access Controls	126
<i>The Importance of Naming</i>	127
The Internet Protocol (IP)	128
<i>Probing Network Paths</i>	131
Problems at the IP Layer	132
<i>Are Your Mission-Critical Applications Safe from Attacks?</i>	135

IPsec	138
Supporting Protocols for IP	139
Address Resolution Protocol (ARP)	139
Domain Name System (DNS)	140
Routing Interchange Protocol (RIP)	141
User Datagram Protocol (UDP)	141
Port Security	142
UDP Security Concerns	142
Transmission Control Protocol (TCP)	142
TCP/IP Security Concerns	143
TCP/IP Application Security	145
Trusted Hosts	145
The Role of the Firewall in Traditional Security	146
What Is a Firewall?	146
Packet Filters Provide Access Control Services	147
Application Proxies Provide Access Control	149
Firewalls Provide IP Security	150
IP Sec or Application Security	151
How Complex Is Your Network Security?	151
Why Intrusion Detection Is Needed after Network Security	153
Part 2 Intrusion Detection: Beyond Traditional Security	155
5 Intrusion Detection and Why You Need It	157
Do You Have Protection?	157
The Role of Intrusion Detection	162
Beyond I&A	162
Beyond Access Control	163
Beyond Network Security	164
Intrusion Detection: Concepts and Definitions	169
IDS Engine Categories	170
Real Time or Interval Based	173
Data Source	174
A Generic IDS Model	176
Getting Ready to Look for Hacker Trade	178
6 Detecting Intruders on Your System Is Fun and Easy	181
Classes of Attacks	182
Internal Attacks	182
External Threats	186
Layers of Information Sources	188

38	<i>Warning: Opportunities for Hackers!</i>	188
39	Commercial IDS Layering	191
39	How Does One Get the Data?	193
40	<i>Intrusion Detection Inside a Firewall</i>	194
41	Relying on Others for Data	194
41	System Data Sources	195
42	syslog	195
42	Audit Trails	198
42	Tracing the Path of Activity Can Be Difficult	200
43	<i>Monitoring Policies</i>	201
45	Simple or Complex Attacks	206
45	Prepare to Scan for Weaknesses	207
46	7 Vulnerability Scanners	209
46	What Is a Scanner?	209
47	Characteristics of Scanners	210
49	Local Scanners	211
50	Remote Scanning	212
51	How a Scanner Works	213
51	Improving Your Security with Scanners	214
53	ISS SAFESuite	214
55	Other Scanners	223
57	Ballista	224
57	IBM Network Security Auditor	224
57	<i>Keeping the Scanners Current</i>	225
62	Are You Done Yet?	225
62	8 UNIX System-Level IDSs	227
63	Detecting Hacks with Stalker	228
64	Audit Management	229
69	Tracer/Browser	230
70	Misuse Detector	231
73	Attacks Detected by Stalker	232
74	Is Stalker Right for You?	233
76	Some Alternative Stalker Configurations	234
78	Detecting Hacks with the Computer Misuse Detection System	235
81	How CMDS Works	236
82	Other IDS Features to Consider	240
82	Ease of Set Up	240
86	Distributed Intrusion Detection	241
88	Monitoring and Privacy	242

Finding New Attacks	243
General Event Monitoring or Intrusion Detection	244
Using Audit Logs to Find Attacks	244
Two Main Reasons for Vulnerabilities	245
Notation	246
A Word about Sequences	247
Focusing on Local Attacks	248
<i>An IDS Limitation</i>	248
<i>The Scope Problem and Memory Requirements</i>	255
Why You're Not Finished Yet	261
9 Sniffing for Intruders	263
How Network IDSs Work	263
Networks and Subnets	263
Network IDSs Sniff Network Traffic	264
Other Network IDS Features	265
Network IDS Attack Recognition	267
<i>Fragmented IP Packets</i>	268
Advantages of Network IDSs	268
Limitations of Network Packet Sniffing	270
Network Sniffers Do Not See All Packets	270
Network Sniffers Are Blinded by Encryption	271
Missed System-Level Attacks	272
The Network IDS Is Not the Destination Node	273
<i>Getting around the Encryption Problem</i>	274
Which Product Has the Best Nose?	276
IBM and NetRanger	277
RealSecure	277
Network Flight Recorder	279
Will Intrusion Detection Be Enough?	280
10 Intrusion Detection for NT	283
NT Security Review	283
Sources of Data for NT IDSs	284
NT Event Log	285
Event Records	287
What to Monitor on NT	288
Increased Privileges	288
Impersonation	289
Remote Attacks	290
Local Vulnerabilities	292

Intrusion Detection Products for NT	293
Look for These Features	294
Centrax	294
For Further Thought	300
Part 3 Rounding Out Your Environment	303
11 You've Been Hit!	305
Be Prepared	305
Discovery and Detection	308
Responding to Intrusions	309
Should You Pursue Your Attacker?	311
12 Intrusion Detection: Not the Last Chapter When It Comes to Security	313
Traditional Computer Security	313
The Basic Security Model	314
I&A	314
Access Control	315
Network Security	315
The Rationale for IDSs	316
Types of IDSs	316
Scanners	317
System-Level IDSs	317
Network Sniffers	319
Improving upon IDSs	320
Increase Application-Level Detection	320
Adapt to Changing I&A	321
Support Common Systems Management	322
Simplify Development of Attack Signatures	322
Combine Products	323
Support Integration into Other Products	323
Support Research	323
<i>Self Reference and IDSs</i>	324
Take It Away	326
Appendix Hot Links for Information	327
Incident Response Organizations	327
Intrusion Detection Research	327
References	329
Index	333

Chapter 9

Sniffing for Intruders

This chapter describes how you can catch intruders by watching the network traffic at your site. From the title, you probably thought you were going to learn how to really put your nose to the trail and smell for hackers. On the other hand, perhaps you envisioned that you might pick up a good bloodhound and track down perpetrators through the electronic jungle. Luckily, no vendor is pushing either of these sniffer approaches. The IDSs described in this chapter work by monitoring network traffic as it flows across subnets at your site.

The material in this chapter is broken into two main sections. First, you'll look at how network IDSs work and examine their pros and cons. Next, some popular network IDSs are described to give you a glimpse of what to expect from leading vendors in this field. Like many of the chapters so far, as you get near the end of this chapter, you will find yourself wondering whether the use of scanners, system IDSs, and network IDSs is finally enough to solve your intrusion detection problems. You probably can guess the outcome.

How Network IDSs Work

What makes a network IDS different from other tools about which you've read? The next few sections describe the basic approach to network intrusion detection. After this, you'll see some example attacks that can be discovered by looking at network packets.

Networks and Subnets

Networked environments usually are divided into multiple subnets for various reasons. By separating computers into physically distinct groups, a network administrator can regulate the flow of traffic into and out of subnets. Gateways

and routers control the boundaries of subnets and decide how packets are delivered to each subnet. The primary advantage of subnetting is performance.

For example, if all of the accounting computers are on the sixth floor of a building, physically configuring a separate accounting subnet will limit accounting network traffic to that floor. If other floors do not need to see these packets, why bother routing the network traffic to them? The degenerate case in which all computers are on the same subnet is undesirable because all network traffic flows on the same network. Subnets usually are set up according to departmental organizations, physical buildings, or security boundaries. Therefore, another advantage of subnetting is security.

A router can be used to block the flow of packets from one network to another. The simplest type of packet filtering firewall is a router. Other uses of screening routers were identified in Chapter 4, "Traditional Network Security Approaches," such as blocking IP address impersonation attempts into your perimeter network.

Naturally, some packets will cross subnets as computers communicate with each other or try to reach computers outside the company's firewall boundary. Sometimes, a subnet is completely disjointed and communicates with no other subnets. Examples include test environments in many companies. Test coordinators prefer to limit the damage of programs that are not ready for the production environment by restricting the network test environment.

As you can see, in a typical enterprise network, many different subnets are self-contained. Network packets may uniquely flow within a subnet and not cross into any other subnet. If you wanted to monitor network traffic in subnets for performance, each subnet would need a separate monitor to watch its unique packets. Why? Simply put, not all subnet traffic is available from a single point on the network. If you really want to watch what everyone on the network is doing—including both insiders and outsiders—then your monitors need to pick up all of the packets. If all of your computers are on a single network, a single monitor is enough. Logically, if you have a network divided into subnets, then you need a monitor on each subnet.

Network IDSs Sniff Network Traffic

Network adapters can be configured to run in *promiscuous mode*, which allows the adapter to grab all of the packets that it sees on the network. The default behavior of a network adapter for a computer is to grab only those packets destined for that computer (sounds like a reasonable performance enhancing choice). In promiscuous mode, the adapter grabs all packets that it senses on the subnet and passes them up to the device driver. The captured packets are passed from the device driver up to the IDS for analysis.

NetRanger developed by the WheelGroup and now owned by Cisco, has a slightly different setup. In its original form, NetRanger works with routers provided by Network Systems Corporation. The router provides network traffic information to NetRanger software running on a designated computer. Communications between the computer and the router are cryptographically secured for privacy and integrity across a socket connection. One advantage of this architecture is that the monitoring node running NetRanger and the router do not need to be on the same network. The two components of the solution need to be reachable only via some type of network, such as via a private dial-up line. A standalone version of NetRanger that uses a computer with a promiscuous mode adapter is also available now. As you might expect, NetRanger also works quite well with Cisco routers, too.

To be effective at catching intruders, network IDSs need to be positioned properly in the network. In order to catch inbound packets for a subnet, the network IDS must be positioned as the first node after the router in the subnet. Alternatively, the IDS may be placed on a gateway between two subnets to watch for attacks across the subnets. A common placement for network IDSs is immediately after the firewall in an enterprise. Because all inbound and outbound traffic must pass through the firewall, the network IDS easily can reside directly after the firewall and as the first node inside the secure network as shown in Figure 9.1.

You should be sure to understand that *the network IDS does not intercept and hold packets* before forwarding them to the intended node. Because the IDS's network adapter is capturing packets in promiscuous mode, it is merely reading the packets as they appear, not grabbing them and holding them until they are analyzed. A network IDS should not interfere with the performance of your network.

Even though the network IDS does not alter your network performance, you should have some performance concerns about the IDS itself. Because it is potentially processing packets for a large number of nodes in your network, the IDS must scale well or else it will drop packets and miss attacks. Before you plop down large amounts of money for a large-scale network IDS solution, insist on performance testing to ensure that the solution meets your needs. You may be able to get around scalability problems by deploying multiple IDSs in a chain and configuring each one to look for a disjointed set of attacks.

Other Network IDS Features

Like any other application that you run in a distributed environment, a network IDS needs to provide useful systems management capabilities. Features needed include the following:

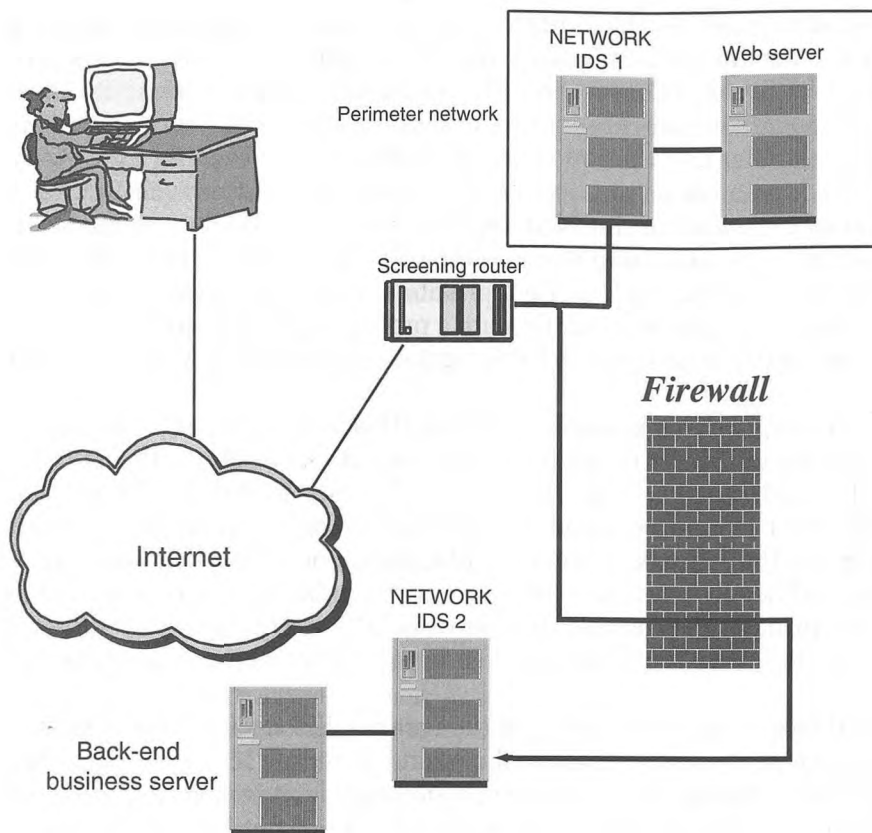


Figure 9.1 A properly positioned IDS will see all of the inbound and out-bound packets.

- ◆ Configuration of multiple network IDSs from a central console
- ◆ Centralized reporting from network IDSs to a central console
- ◆ Secure communications between distributed components
- ◆ Configurable sets of attack signatures to monitor
- ◆ Easy-to-read reports
- ◆ Real-time alerts and optional responses
- ◆ Integration with system management frameworks, such as Tivoli TME

Depending on your level of sophistication, you might also be interested in designing your own attack signatures. Not all network IDSs provide a simple way for you to create signatures.

Network IDS Attack Recognition

What can a network IDS detect? First, because the source of information is network packets, network IDSs look for *attacks that are targeted at network protocols*. Examples include Ping of Death and SYN Flood because both of these are attacks against weaknesses in TCP/IP itself. Problems in other protocols, such as Novell IPX and Microsoft SMB, also are found.

Protocol problems result in other attacks such as the following:

- ◆ Sequence number guessing attempts
- ◆ IP address impersonation
- ◆ Session hijacking
- ◆ IP fragmentation
- ◆ Other well-known denial-of-service attacks (the “Pizza” attack mentioned earlier in the book)

Next, by analyzing packet data content (as opposed to header fields) a network IDS can look for attacks such as the AIX “rlogin -froot” bug. Other example *application attacks and vulnerabilities* detected by network IDSs are as follow:

- ◆ Various sendmail bugs (EXPN, VRFY, debug)
- ◆ phf, test.cgi, and other CGI bugs
- ◆ Buffer overflows in finger and DNS
- ◆ Various NFS, FTP, and TFTP bugs

Unlike scanners that occasionally probe your systems for these weaknesses, network IDSs look for evidence of someone mounting one of these attacks against your systems in real time. The evidence is found by inspecting the contents of packets.

Finally, scanning packet data for unauthorized strings such as “confidential”, “proprietary”, “secret”, and other *potential leakages* is another feature found in some network IDSs. A list of keywords can be configured into the IDS. Naturally, performance concerns arise if the list is too long.

Fragmented IP Packets

Breaking down packets into smaller chunks and reassembling them into the proper format is something that happens often in network communications. You saw how this happens between network layers on the same system, and between peer layers on different systems when you read Chapter 4, "Traditional Network Security Approaches."

A well-known attack called IP Fragmentation tries to inject or form bogus IP packets so that when they are reassembled at the target node, there is a chance for a successful hack. Various operating systems handle reassembly of fragmented packets differently. So, the attack will not always succeed.

Recall that sequence numbers are included in TCP/IP packets so that the receiving node can reassemble packets received out of order into the proper format. By messing around with sequence numbers in injected or forged packets, an attacker can trick the receiving system into overlaying already received data with something else. For example, if the target node received a packet with the sequence number for bytes 1–5 and data "smith." The hacker could send another packet with the same sequence number but data "root." Depending on how the receiving OS handles this condition, it will either overlay the first packet or discard the new (hacked) packet. The behavior of the OS determines whether the hack succeeds or not.

Chapter 7, "Vulnerability Scanners," described useful scanner tools that can be used to probe your systems for weaknesses. If a hacker is scanning your system with ISS or SATAN, a network IDS should be able to detect the activity. Of course, looking for patterns such as port scanning of TCP/IP ports is a process that also is affected by time. What if an attacker scans one port per day? Is this something that a network IDS can detect? Most network IDSs have a time-out setting that you can configure to determine the interval, which constitutes a group of related events such as sequential port scans. If you set this value too small, you could miss sequential port scans across days, for example.

Advantages of Network IDSs

One of the main advantages of a network IDS is simple implementation. Unlike system-level intrusion detection, which requires a monitor to be running on

every system, network IDSs require one monitor per subnet. Reduced cost is one consequence of this feature. Installing a single network IDS should be cheaper than installing client system level monitors on each node. In some cases, you might want to run a network IDS monitor on each of several nodes in your environment. Most network IDS architectures support this configuration today.

Now you could get really picky and claim that system-level IDSs could gather the data from each system and then forward it to a central analyzer. However, the real issue is that system-level monitoring requires you to *gather* information from each system by running some type of sensor or monitor on each system. A network IDS gathers information by actively monitoring network traffic without requiring a separate sensor on each system. Of course, network IDSs cannot detect some of the intrusions and misuses that system IDSs can, and vice versa. You'll see the limitations in the next section.

Another advantage of network IDSs is that the data which they gather comes essentially for free. Computers are emitting network traffic as part of the normal routine of communicating between each other. The network IDS needs only to attach to the network and sniff this information as it appears. A network IDS is noninvasive because it does not alter in any way the systems you want to monitor. None of the system calls in the kernel are modified or replaced on any systems in the network (with the possible exception of the network IDS node itself). Nor does a network IDS require you to introduce a new data source, such as audit logs or syslog. System-level IDSs, as noted in Chapter 6, "Detecting Intruders on Your System Is Fun and Easy," may require you to turn on auditing or syslog in order to capture activities on the system. If you already are running the audit subsystem to track system activities, this practice should not bother you. However, if auditing and syslog are not running on your systems today, a network IDS is appealing.

Perimeter security is what the network IDS is primarily designed to monitor. As more companies connect into cyberspace, increasing threats from intruders are inevitable. Network IDSs aim to simplify the task of monitoring network traffic for security violations and intrusions. Because the amount of network traffic generated by an enterprise can be tremendous, having a system that automatically looks for problems and responds to events is necessary. Note that this type of IDS is a logical extension of network performance monitoring with automated responses.

Many system-level IDSs do not have ample data to detect network intrusions or misuses. Neither the audit logs nor syslog give detailed information about network packets. To get at the content of the packets themselves, the IDS needs to do the following:

- ◆ Run as part of the OS and analyze every packet that arrives or leaves the node
- ◆ Run on a separate node that monitors network traffic for all nodes

The latter approach seems to be the most scalable today. Limitations of separate node network IDSs may force administrators to run a network IDS on each node in the future.

Network IDSs usually are equipped with some form of response or countermeasure feature. NetRanger can send commands to the router to block packets from a particular source IP address when attacks originate from that address. RealSecure and other stand-alone monitors can send *block address* commands to popular firewalls, too. One already mentioned danger of these countermeasures is that frequently the hacker is using forged addresses. You could end up blocking your biggest Web site customer if suddenly a hacker forges a SYN Flood attack from that customer's IP address.

Limitations of Network Packet Sniffing

Although network IDSs are an essential weapon in the security officer's arsenal, it's important to understand their limitations. The following sections identify problems with network IDSs so that you can understand what to expect from them when in use at your site.

Network Sniffers Do Not See All Packets

A network IDS works by running a network adapter in promiscuous mode to capture all of the packets coming into and going out of a particular subnet. *Notice that this is not the same as watching all of the network traffic that appears on a subnet.* Look at Figure 9.2. Here, the physical arrangement of the nodes is in a ring with node B sitting between node A and the node running the IDS. The packet "Hello B" is sent from node A to node B. However, because A and B are directly adjacent, B grabs and processes the packet sent by A. The node running the IDS never has a chance of seeing the packet.

This means that a network IDS is not designed to track all the network activities on a subnet. Instead, the IDS is positioned to look for inbound and outbound packets at the entry/exit of the subnet. Following terminology introduced earlier in the book, the network IDS catches intruders, but it does not always catch internal misuse. If the packet from A to B had been a misuse or internal hack, the IDS

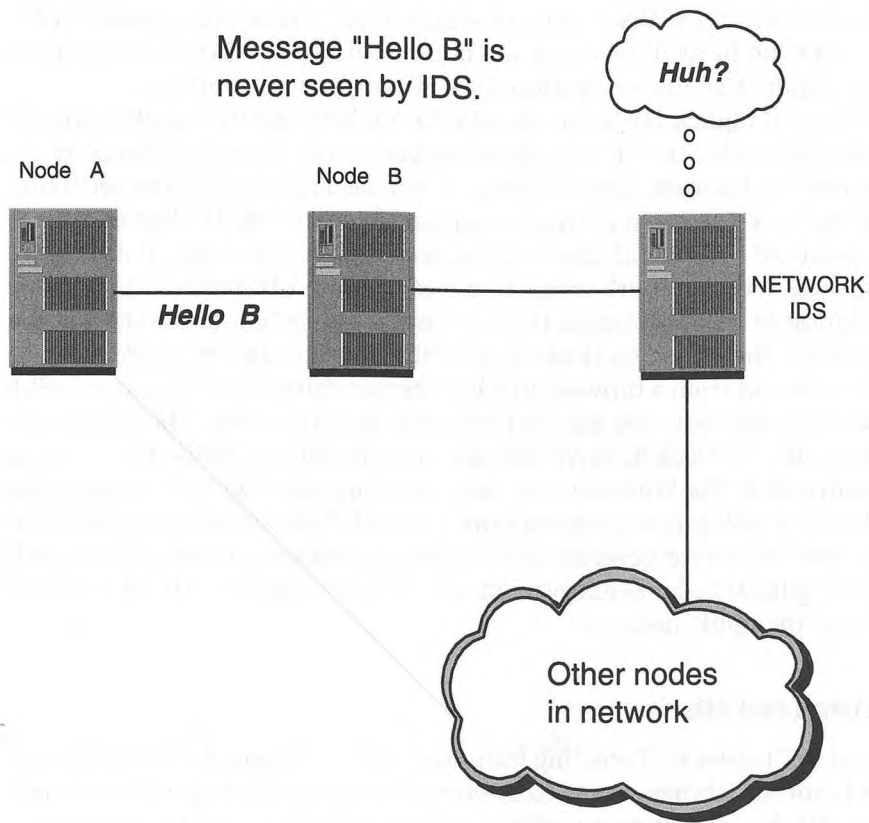


Figure 9.2 An IDS does not see all packets on a subnet.

node would miss it. To catch attacks between nodes, an intelligent IDS sniffer would need to be run on each node.

Network Sniffers Are Blinded by Encryption

Many sites rely on encryption for privacy of network traffic. In some cases, two corporate sites are connected by an IP tunnel. A firewall at each site implements the IP tunnel so that all traffic is encrypted as it passes across the unsecure Internet. After a firewall receives an encrypted packet from another site, the packet is decrypted and sent on to the target node in the secure network. A configuration

like this *does not* hinder network intrusion detection. The packet appears in the clear as it leaves the firewall. Because the network IDS is the first node after the firewall (see Figure 9.1), the encryption does not impact the solution.

In some cases, though, an IP tunnel is established between two *arbitrary* nodes in a network. The nodes could be in the same subnet, or they could be communicating across the Internet. The IP traffic is not decrypted until the receiving node reads the packet from its network adapter. The network IDS has no way of seeing the cleartext version of the packets. Any attack signatures that require cleartext packets will not work when two nodes use an IP tunnel. Again, one possible solution to this problem is to run a sniffer on each node. Note that the sniffer must be in the OS network stack *after* the packets are decrypted.

When you connect from a browser to a Web server using *secure sockets* (SSL), the packets from your computer are not decrypted until they reach the Web server application itself. SSL packets flow through the firewall and remain encrypted. The packet arrives at the Web server node, moves up through the kernel stack, and is read by the Web server program from a socket. It is not until this last step, which only the Web server program itself controls, that the packet is decrypted. This type of application-level encryption also blinds network sniffers to many attacks such as the “phf” hack.

Missed System-Level Attacks

As mentioned in Chapter 6, “Detecting Intruders on Your System Is Fun and Easy,” system-level monitoring has access to important events such as privilege transition. A *new* attack that causes a buffer overflow and gives root privileges to a remote user will not be seen by a network IDS. If the attack signature is written properly, the system-level IDS will detect and respond to this type of situation.

Two general classes of attacks exist that a network IDS cannot detect, but a system-level IDS can. You can think of the first class as *unknown side effects*. When an activity on the system happens as the result of receiving a network packet, it’s possible that a side effect will occur that violates your security policy. Examples include the following:

- ◆ Creation of a world-writable file by a privileged program as a result of processing a network packet
- ◆ Downgrading the security of an existing resource, such as making `/etc/passwd` world writable
- ◆ Upgrading the privilege of a user, such as changing the UID of a normal user to zero in `/etc/passwd`

- ◆ Creation of a back door, such as any program that can lead the user to a root shell

Unless the hacks that led to these breaches already are known in the security community, the network IDS will not see these events, but the system-level IDS will. If you have a scanner, some of these problems will be caught the next time it runs. In some sense this argument seems unfair because it merely states that if the attack is not known in the community, the network IDS vendor cannot build a signature to catch the attack. However, even if the initiation sequence for the attack is unknown, a system-level IDS *can detect* that a SUID root program was created. What this says to you is that you need both types of IDSs—system and network—to catch all of the attacks you face.

The other class of system-level problems that a network IDS misses *are attacks that are not based on sending or receiving network packets*. Examples include any hacks launched by directly attached terminals or TTYs. If you are connected to the computer system with a terminal, you can start a nasty brute force password guessing program, and no network sniffer will be able to detect it. Most midrange hardware vendors still sell a significant number of dumb terminals to customers. Naturally, these threats are posed mostly by insiders rather than intruders.

The Network IDS Is Not the Destination Node

Recently, Ptacek and Newsham (1998) identified several weaknesses in intelligent network sniffers. Similar concerns about network IDSs were simultaneously identified by Paxson (1998). At the heart of the discussion is the fact that a network IDS cannot know for sure what is happening on the network nodes themselves. An inbound packet has a destination IP address for the intended recipient. The network IDS does not know for sure whether the destination node will accept a packet or discard it. To really know the behavior of the remote node, the network IDS must contain a good deal more knowledge than it does today.

For example, operating systems handle fragmented packets differently. Some OSs will discard a packet if it contains overlapping sequence numbers with previously received packets. Other OSs will accept the packet and process it; thereby overlaying existing data received earlier.

One example given by Ptacek and Newsham is the checksum on a packet. Most systems will discard a packet with a bad checksum, but some network IDSs do not currently check this as part of packet inspection. Now that the issues have been publicly indicated, you can expect future IDS releases to include this feature. Because the IDS is *not* the real recipient of the packet itself, the IDS cannot

Getting around the Encryption Problem

The encryption issue is a particularly sticky one for network IDSs. It is highly likely that over time, you will see more encryption of application-level data. You need encryption for secure communications. There isn't much you can do to change the limitations on network IDSs when encryption hides the network packet content. IDS vendors do have a few alternatives, though.

Instead of spelling the demise of network IDSs, encryption argues for a repositioning of the technology. If a large number of applications rely on SSL, for example, the SSL libraries could be enhanced to invoke network IDS routines *after* the packets are decrypted. *The added advantage is that the IDS routines also would be running on every node that uses SSL, thus providing previously unavailable opportunities for misuse detection.*

A similar approach would be to embed the network IDS routines directly in the network stack of the operating systems. As data from a socket is read by the application, the IDS routines optionally could be called to look for attacks. A socket option that each application could set would provide granular control over when to invoke the IDS routines. A system-wide switch to enable the IDS for all socket reads is another configuration possibility. The IDS routines would be running on each node with the advantage for misuse detection again.

Repackaging network IDSs as a set of library APIs is another possibility. As applications read data from sockets, they optionally could call the IDS routines to check for problems. This is a slight variation on the socket option suggested above. One difference is that it would be easier for a vendor to market network IDS libraries than it would be to convince a number of OS providers to embed the code. Unfortunately, if the APIs are optional, it's difficult to encourage programmers to modify existing software to take advantage of the libraries. When the IDS routines are a default option in the network stack of the OS, the solution is more transparent and easier for application programmers to use.

know for sure how the destination node will really handle the packet. Many factors, such as memory limitations, determine whether a packet is dropped or processed by the intended receiver. The IDS cannot possibly know all of these variables for each node on the subnet it monitors.

This subtle point—not knowing for sure how the destination node will respond—can lead to some interesting attacks. In an *insertion* attack, an adversary will inject a packet that the IDS will accept but which the destination node will reject or drop. The IDS and the destination node are thus *in different states* because they are processing different data. A few clever hacks show how to sneak a phf (or similar) attack past current network IDSs even though they will be executed successfully on the destination machine. As a teaser to completely read the referenced papers, think about how you could send a string such as “ppppphh-hhhhhhhhhhhh?rrrrmmmm *”. You could send individual one-byte packets formed in a such a way that the destination node would drop all packets except those that combine into “phf?rm*”.

You also can inject SYN packets that can trick the IDS into resetting its state, even though the target node ignores the packet. When the IDS resets its state, pattern matching that was in progress is restarted; thus, losing any attacks in progress for that TCP session.

In an *evasion* attack the destination node accepts a packet that is ignored by the IDS. For example, the attacker could send extra packets with the same sequence number as a previous packet but with different data. The IDS might drop the packet because the sequence number was already used. The destination node (depending on the OS) would accept and process the packet, replacing a previous substring with one that turns the entire message into a hack. Many other evasion attacks are possible because of protocol problems with IP, TCP, or UDP.

Another problem to watch for is denial-of-service launched against the IDS itself. While sniffing network traffic, the IDS maintains a queue for incoming packets. The amount of memory that is allocated for these queues often can be configured, but it is eventually bounded by some constraint—whether physical memory or virtual memory. If a hacker knows what the IDS is looking for, it's not difficult to inject a number of packets that must be processed. The IDS can run out of resources.

Recall that network IDSs operate in passive mode and sniff the network. They do not block packets, which is one thing that firewalls do for you. When a firewall fails, it (generally) does not let any packets through, and thus your security policy is not violated. If a network IDS fails due to resource exhaustion, your policy can be violated because the IDS does not block packets; it works by alerting you when there is a problem.

In this section, you've seen a number of problems with network IDSs. Don't let this discussion give you the impression that these IDSs are weak products. No security tool is perfect. Network IDSs are critical for improving security at your site. These IDSs also are very useful for catching a number of attacks against your

network. Vendors have been responsive to concerns such as those mentioned previously and have speedily added fixes in new releases.

Which Product Has the Best Nose?

An *Infoworld* test reported in the May 4, 1998 issue rated products as follows:

1. IBM's outsourced solution using NetRanger
2. ISS Real Secure
3. Network Flight Recorder (NFR)
4. Abirnet Session Wall

The study by the *Infoworld* team announced a suite of 16 well-known network attacks that they tried against the products. Only NFR caught all of the attacks. The team used the scripting language, with help from Anzen, to build tests that do the following:

- ◆ Probed for information, tried to gain access
- ◆ Launched denial-of-service attacks
- ◆ Attempted to overburden the IDS with a combination of the preceding

The chosen IDS were challenged with attacks such as the following:

- ◆ Ping of Death
- ◆ SATAN scanning
- ◆ ISS SAFESuite scanning
- ◆ Port scanning
- ◆ ftp cwd ~root
- ◆ phf
- ◆ SYN Flood

In all, 23 attacks were attempted individually, with two combinations of attacks completing the full suite of 25. (Some of the 16 attacks have more than one variation that is how one arrives at 23 individual attacks.)

A three-way tie for first place exists between IBM/NetRanger, RealSecure, and NFR. Abirnet fell into last place for three main reasons—it lacks systems man-

agement; it does not have specific IDS reports; and it failed to detect 7 out of 25 attacks. The reviewers nonetheless liked many of SessionWall's features. The next few sections focus on the three IDSs that tied for first place.

IBM and NetRanger

As noted previously, NetRanger is a passive network monitor that is offered with an NSC router or as a stand-alone product on a UNIX box. Hierarchical secure remote reporting between sensor stations and a console is one of the key features of NetRanger. The WheelGroup also reports that NetRanger is more scalable than any other network IDS. NetRanger also can detect session hijacking—something that other network IDSs do not claim.

NetRanger not only detects events but also responds to them as well. Shunning IP addresses for an interval of time is one of the operations that NetRanger can send to the NSC router if you are running that combination. As you might expect, a wide range of response options are available including pager notification, e-mail, and pop-up alerts. Logging and reporting are standard features.

NetRanger allows scanning for administrator-defined strings in network packets—a feature that other network IDSs must soon provide. However, it is not a trivial task to add your own attack signatures to those already supported by NetRanger. This shortcoming is shared by many IDSs.

IBM's Emergence Response Center offers a fee-based service with NetRanger. Instead of staffing your own team of security experts, you can use IBM's strength in this area. A network operations center is staffed 7 × 24, and a specific expert is assigned to your account. When an event is detected, IBM's security experts notify you and help you respond to the event. Up-front planning and response policy design also are available. As hackers become more sophisticated, outsourcing your network intrusion detection seems attractive because you may not be able to staff and maintain your own center of competency.

One final note about NetRanger is worth mentioning. Some of the founders and technical leads for the WheelGroup have worked at the Air Force Warfare Information Center and at the NSA. With contacts like that, it's not surprising that a number of government sites depend on NetRanger for network intrusion detection. You know NetRanger has been tested substantially in the field.

RealSecure

ISS is already the market leader in scanning tools with SAFESuite. RealSecure is a widely used network IDS that complements ISS's other offerings. Like NetRanger, RealSecure supports remote sensing stations, called *engines*, that



Figure 9.3 RealSecure's initial management panel.

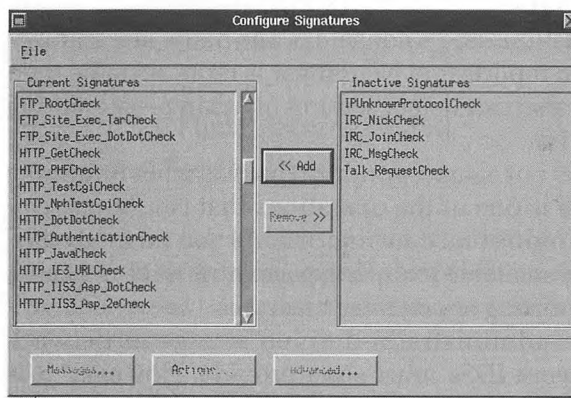


Figure 9.4 RealSecure attack signature configuration.

report to a central console. Naturally, communication between engines and the console are cryptographically protected using a shared pass phrase. Figure 9.3 shows the initial panel for RealSecure.

Monitoring and response options can be customized for each engine. Recall from the previous discussions that your site should have one monitoring engine per subnet (possibly more for performance gains). In Figure 9.4, you see some of the attack signatures that can be configured by node in RealSecure. As before, a comprehensive list of attacks detected is best obtained from ISS because the product is updated regularly.

RealSecure also supports a playback mode, which can be used to dig through the network traffic looking for problems. In playback mode, the product does not run attack signature recognition on the playback traffic. This feature probably will be fixed in the near future. Once activated, the console begins receiving data from the remote engines. You can choose from a number of different views on the console including by node or by event severity (high, medium, or low). Data from the engines is logged, and a variety of reports are possible.

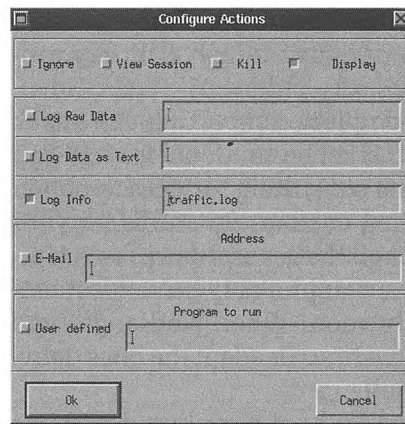


Figure 9.5 RealSecure response configuration.

A number of different response options are available, including killing the offending network connection by sending a RST packet. Figure 9.5 gives a snapshot of how one might configure response options in RealSecure. *Templates* that declare signatures to use and how to respond to events can be applied to different engine nodes. Notifying an administrator is supported as a response along with the more aggressive socket kill option.

RealSecure runs on UNIX and NT platforms. The engines and console can run on different OS platforms, too. Regardless of the platform on which the engine is running, it can detect specific attacks against TCP/IP, NT, NETBIOS, and UNIX. For example, even if the engine is running on an NT workstation, it can detect someone trying to exploit the old AIX “rlogin -froot” bug.

ISS also relies on its X-Force team of security experts to find new attacks and create (or adjust) signatures. Discoveries can come from the X-Force’s own research or from contacts that it has with the underground. ISS is well known for its NT expertise, with Microsoft often working closely with X-Force team members.

Network Flight Recorder

The interesting feature of NFR is that it is not designed to be an IDS. Instead, NFR is a general-purpose network monitoring tool. NFR just happens to include a general-purpose scripting language that can be used to build attack signature recognition routines. Note that Checkpoint adheres to this philosophy in the

firewall market by offering its INSPECT language for similar reasons. The argument, and it is a reasonable one, is that passing packets through arbitrary programs provides great flexibility in enforcing a security policy.

As a security officer, knowing that you have complete control over the attack signatures is a compelling thought. However, you must be cautious because you could make a mistake in programming. Relying on consulting services to obtain signatures for well-known attacks, and developing custom signatures for your proprietary applications is a good tradeoff. Although the general-purpose scripting language in NFR is its most appealing feature, you probably can expect network IDS vendors to offer similar functionality soon. When NetRanger and RealSecure enables you to easily add your own signatures or adapt the ones delivered with the tools, the appeal of NFR will be challenged.

Earlier versions of NFR did not have the same level of distributed systems management provided with NetRanger and RealSecure. This level probably is supported in the version on the market today. Paging is a standard notification mechanism supported, and the summary reports are exceptional. If you invest in NFR, you'll also be able to use expertise to watch your network behavior in other ways, such as monitoring performance.

Will Intrusion Detection Be Enough?

It would be wonderful if this chapter could close by claiming victory in the war on intruders. You know by now that perfect security is impossible. You've had a chance to see how scanners, system-level tools, and network IDSs are able to catch some hacks but miss others. Your job is to know the types of IDS tools that are available, know what they *can* do, and know what they *cannot* do in order to properly rely on them for improving your security. The bad news is that no single type of IDS today will be sufficient by itself. The good news is that you can buy several tools that *do* give you ample coverage against attacks. Today, not all of the tools will come from the same vendor, nor is it likely that they will interoperate. This situation is changing, though.

Vendors who build each type of tool make design tradeoffs when building a scanner, system, or network IDS. In the recent past, a vendor might focus on one or two of these tools types but not offer solutions covering all three categories. Alliances, acquisitions, and new product offerings by IDS vendors are becoming more inclusive. In 1998, you should see some significant improvements in this area. The final chapter of the book offers some suggestions for what you can do today, and what you can expect from IDSs in the future.

Another issue you will need to contend with is increasingly sophisticated hackers. IDS vendors try to keep abreast of new hacks and modify their tools to detect these. However, a lag always exists between a clever new way to break into a system and the products that try to find hackers.

Much of the material on intrusion detection so far has focused on UNIX systems and TCP/IP. This focus was used to keep the discussions simple. Including NT and UNIX comparisons in each section would have been too confusing. Now that you're an expert on IDSs and understand how they apply in UNIX environments, you are ready for the next chapter, which examines intrusion detection for NT.