Network Security Hacks

By Andrew Lockhart

Publisher :	O'Reilly
Pub Date :	April 2004
ISBN :	0-596-00643-8
Pages :	312
Slots :	1.0

This information-packed book provides more than 100 quick, practical, and clever things to do to help make your Linux, UNIX, or Windows networks more secure. Loaded with concise but powerful examples of applied encryption, intrusion detection, logging, trending, and incident response,

Network Security Hacks demonstrates effective methods for defending your servers and networks from a variety of devious and subtle attacks.

- Table of Contents
- Index
- Reviews
- Reader Reviews
- Errata
- Academic

Network Security Hacks

By

Andrew Lockhart

Publisher : O'Reilly
Pub Date : April 2004
ISBN : 0-596-00643-8
Pages : 312

Slots : 1.0

Credits

About the Author

Andrew Lockhart is originally from South Carolina but currently resides in northern Colorado, where he spends his time trying to learn the black art of auditing disassembled binaries and trying to keep from freezing to death. He holds a BS in computer science from Colorado State University and has done security consulting for small businesses in the area. When he's not writing, he currently works at a Fortune 100 company. In his free time, he works on Snort-Wireless (http://snort-wireless.org), a project intended to add wireless intrusion detection to the popular open source IDS Snort.

Contributors

The following people contributed their hacks, writing, and inspiration to this book:

- Oktay Altunergil is the founder of The Free Linux CD Project (http://www.freelinuxcd.org) and one of the maintainers of Turk-PHP.com (a Turkish PHP portal). He also works full-time as a Unix system administrator and PHP

  programmer.

- Michael D. (Mick) Bauer (http://mick.wiremonkeys.org) writes Linux Journal's "Paranoid

  Penguin" security column. By day, he works to keep strangers out of banks' computer networks.

- Schuyler Erle (http://nocat.net) is a Free Software developer and activist. His interests include collaborative cartography, wireless networking, software for social and political change, and the Semantic Web. Schuyler is the lead developer of NoCatAuth, the leading open source wireless captive portal.

- Bob Fleck (http://www.securesoftware.com) is Director of Security Services at Secure Software. He consults in the fields of secure development and wireless security, and is a coauthor of O'Reilly's *802.11*

  Security book. The results of his more recent investigations into Bluetooth security can be found at http://bluetooth.shmoo.com.

- Rob Flickenger (http://nocat.net) is a writer and editor for

  O'Reilly's Hacks series. He

  currently spends his time hacking on various projects and promoting community wireless networking.

- Michael Lucas (http://www.blackhelicopters.org/~mwlucas) lives in a haunted house in Detroit, Michigan, with his wife Liz, assorted rodents, and a multitude of fish. He has been a pet wrangler, a librarian, and a security consultant, and now works as a network engineer and system administrator with the Great Lakes Technologies Group. He's the author of

  Absolute BSD, *Absolute OpenBSD*, and *Cisco Routers for the Desperate*, and is currently preparing a book about NetBSD.

- Matt Messier (http://www.securesoftware.com) is Director of Engineering at Secure Software and a security authority who has been programming for nearly two decades. In addition to coauthoring the O'Reilly books *Secure Programming Cookbook for C and C++* and *Network Security with OpenSSL*, Matt coauthored the Safe C String Library (SafeStr), XXL, RATS, and EGADS.

- Ivan Ristic (http://www.modsecurity.org) is a web security specialist and the author of `mod_security`, an open source intrusion detection and prevention engine for web applications. He is a member of the OASIS Web Application Security Technical Committee, where he works on the standard for web application protection.

- John Viega (http://www.securesoftware.com/) is Chief Technology Officer and Founder of Secure Software. He is also the coauthor of several books on software security, including *Secure Programming Cookbook for C and C++*

  (O'Reilly) and *Building Secure Software* (Addison-Wesley). John is responsible for numerous software security tools, and he is the original author of Mailman, the GNU mailing list manager.

Acknowledgments

I would first like to thank the illustrious DJ Jackalope (aka Karen) for all of her encouragement, support, and understanding throughout the writing of this book. Without her, it would have languished in the doldrums, and I don't think I could have done this book without her.

I'd also like to thank Nat Torkington for putting me in touch with Rob, my fearless (and patient) editor for this book, as well as my parents for having the faith to let me have time to tinker with computers and do silly things like read

Phrack when I was a kid; if not for that, I might have ended up doing something completely different.

Preface

Nowhere is the term *hacker* more misconstrued than in the network security field. This is understandable because the very same tools that network security professionals use to probe the robustness of their own networks also can be used to launch attacks on any machine on the Internet. The difference between system administrators legitimately testing their own machines and a system cracker attempting to gain unauthorized access isn't so much a question of techniques or tools, but a matter of intent.

After all, as with any powerful piece of technology, a security tool isn't inherently good or badthis

determination depends entirely on how it is used. The same hammer can be used to either build a wall or knock it down.

The difference between "white hat"

and "black hat" hackers

isn't the tools or techniques they use (or even the color of their hats), but their intent. The difference is subtle but important. White hat hackers find that building secure systems presents an interesting challenge, and their security can be truly tested only through a thorough knowledge of how to subvert such systems. Black hat hackers (more appropriately called *crackers*) pursue precisely the same knowledge, but without regard for the people who built the systems or the servers they attack. They use their knowledge to subvert these systems for their own personal gain, often to the detriment of the systems they infiltrate.

Of course, tales of daring international techno-robberies and black-clad, cigarette-smoking, laptop-wielding evil masterminds tend to sell better than simple tales of the engineer who built a strong network, and so the term *hacking* has a bad reputation in the popular press. They use it to refer to individuals who break into systems or who wreak havoc using computers as their weapon. Among people who solve problems, though, the term *hack* refers to a

"quick-n-dirty" solution to a

problem, or a clever way to get something done. And the term *hacker* is taken very much as a compliment, referring to someone as being *creative*, i.e., having the technical chops to get things done. The Hacks series is an attempt to reclaim this word, document the ways people are hacking (in a good way), and pass the hacker ethic of creative participation on to the uninitiated. Seeing how others approach systems and problems is often the quickest way to learn about a new technology.

Only by openly discussing security flaws and implementations can we hope to build stronger systems.

Why Network Security Hacks?

Network Security Hacks is a grimoire of 100

powerful security techniques. This volume demonstrates effective methods for defending your servers and networks from a variety of devious and subtle attacks. Within this book are examples of how to detect the presence (and track every keystroke) of network intruders, methods for protecting your network and data using strong encryption, and even techniques for laying traps for would-be system crackers.

Many important security tools are presented, as well as clever

methods for using them to reveal real, useful information about what is happening on your network.

How This Book Is Organized

Although each hack is designed to stand on its own, this book makes extensive use of cross-referencing between hacks. If you find a reference to something you're interested in while reading a particular hack, feel free to skip around and follow it (much as you might while browsing the Web). The book itself is divided into several chapters, organized by subject:

Chapter 1, *Unix Host Security*

> As the old saying goes, Unix was designed to share information, not to protect it. This old saw is no longer true with modern operating systems, where security is an integral component to any server. Many new programs and kernel features have been developed that provide a much higher degree of control over what Unix-like operating systems can do. Chapter 1 demonstrates advanced techniques for hardening your Linux, FreeBSD, or OpenBSD server.

Chapter 2, *Windows Host Security*

> Microsoft Windows is used as a server platform in many organizations. As the Windows platform is a common target for various attacks, administering these systems can be challenging. This chapter covers many important steps that are often overlooked by Windows administrators, including tightening down permissions, auditing all system activity, and eliminating security holes that are present in the default Windows installation.

Chapter 3, *Network Security*

> Regardless of the operating system used by your servers, if your network is connected to the Internet, it uses TCP/IP for communications. Networking protocols can be subverted in a number of powerful and surprising ways, leading to attacks that can range from simple denial of service to unauthorized access with full privileges. This chapter demonstrates some tools and techniques used to attack servers using the network itself, as well as methods for preventing these attacks.

Chapter 4, *Logging*

> Network security administrators live and die by the quality of their logs. If too little information is tracked, intrusions can slip by unnoticed. If too much is logged, attacks can be lost in the deluge of irrelevant information. Chapter 4 shows you how to balance the need for information with the need for brevity by automatically collecting, processing, and protecting your system logs.

Chapter 5, *Monitoring and Trending*

> As useful as system logs and network scans can be, they represent only a single data point of information, relevant only to the instant that the events were recorded. Without a history of activity on your network, you have no way to establish a baseline for what is "normal," nor any real way to determine if something fishy is going on. This chapter presents a number of tools and methods for watching your network and services over time, allowing you to recognize trends that will aid in future planning and enable you to tell at a glance when something just isn't right.

Chapter 6, *Secure Tunnels*

> How is it possible to maintain secure communications over networks as untrustworthy as the Internet? The answer nearly always involves powerful encryption and authentication techniques. Chapter 6 shows you how to implement powerful VPN technologies, including IPSec, PPTP, and OpenVPN. You will also find techniques for protecting services, using SSL, SSH, and other strong encryption tools

Chapter 7, *Network Intrusion Detection*

How do you know when your network is under attack? While logs and historical statistics can show you if something is out of sorts, there are tools designed to notify you (or otherwise take action) immediately when common attacks are detected. This chapter centers on the tremendously popular NIDS tool Snort and presents many techniques and add-ons that unleash this powerful tool's full potential. Also presented are methods for setting up your own "honeypot" network to attract and confuse would-be system crackers.

Chapter 8, *Recovery and Response*

Even the most competent and careful network administrator will eventually have to deal with successful security incidents. This chapter contains suggestions on how to verify your system's integrity, preserve evidence for later analysis, and track down the human being at the other end of undesirable network traffic.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, daemons, programs, and Unix utilities.

`Constant width`

Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, or the output from commands.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values.

Color

The second color is used to indicate a cross-reference within the text.

The thermometer icons, found next to each hack, indicate the relative complexity of the hack:

🌡 beginner	🌡 moderate	🌡 expert

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the

code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books

does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation

does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Network Security Hacks* by Andrew Lockhart. Copyright 2004 O'Reilly Media,

Inc., 0-596-00643-8."

If you suspect your use of code examples falls outside fair use or the permission given here, feel free to contact us at

permissions@oreilly.com.

How to Contact Us

Please address comments and questions concerning this book to the

publisher:

O'Reilly & Associates
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

We have a web page for this book, where we list

errata, examples, and any additional information. You can access this

page at:

http://www.oreilly.com/catalog/netsechacks

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers,

and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Chapter 1. Unix Host Security

Hacks #1-20

Networking is all about connecting computers together, so it follows that a computer network is no more secure than the machines that it connects. A single insecure host can make lots of trouble for your entire network, as it can act as a tool for reconnaissance or a strong base of attack if it is under the control of an adversary. Firewalls, intrusion detection, and other advanced security measures are useless if your servers offer easily compromised services. Before delving into the network part of network security, you should first make sure that the machines you are responsible for are as secure as possible.

This chapter offers many methods for reducing the risks involved in offering services on a Unix-based system. Even though each of these hacks can stand on its own, it is worth reading through this entire chapter. If you only implement one type of security measure, you run the risk of all your preparation being totally negated once an attacker figures out how to bypass it. Just as Fort Knox isn't protected by a regular door with an ordinary dead bolt, no single security feature can ultimately protect your servers. And the security measures you may need to take increase proportionally to the value of what you're

protecting.

As the old saying goes, *security* isn't a noun, it's a

verb. That is, security is an active process that must be constantly followed and renewed. Short of unplugging the machine, there is no single action you can take to secure your machine. With that in mind, consider these techniques as a starting point for building a secure server that meets your particular needs.

```
# <b>mount -o nodev,noexec,nosuid /dev/hda3 /tmp</b>
```

```
/dev/hda3 /tmp ext3 defaults,nodev,noexec,nosuid 1 2
```

By carefully considering your requirements and dividing up your storage into multiple filesystems, you can utilize these mount options to increase the work that an attacker will have to do in order to further compromise your system. A quick way to do this is to first categorize your directory tree into areas that need write access for the system to function and those that don't. You should consider using the read-only flag on any part of the filesystem where the contents do not change regularly. A good candidate for this might be */usr*, depending on how often updates are made to system software.

Obviously, many directories (such as */home*) will need to be mounted as read-write. However, it is unlikely that users on an average multiuser system will need to run SUID binaries or create device files within their home directories. Therefore, a separate filesystem, mounted with the `nodev` and `nosuid` options, could be created to house the users' home directories. In addition, if you've determined that your users will not need to execute programs stored in their home directories, you can use the `noexec` mount option as well. Similar situations also arise when looking at */tmp* and */var*, where it is highly unlikely that any process will legitimately need to execute SUID or non-SUID binaries or access device files. This helps prevent the possibility of an attacker leaving a Trojan horse in common directories, such as */tmp* or a user's home directory. The attacker may be able to install the program, but it cannot actually run, with or without the proper chmod bits.

Note that services running in a [Hack #10] nodev is specified on the filesystem running under the chroot. This is because device nodes such as `/dev/log` and `/dev/null` must be available within the `chroot()` environment.

There are a number of ways that an attacker can still circumvent these mount restrictions. For example, the `noexec` option on Linux can be bypassed by using `/lib/ld-linux.so` to execute binaries residing on such filesystems. At first glance, you'd think that this can be remedied by making *ld-linux.so* nonexecutable, but this would render all dynamically linked binaries unexecutable. So, unless all of the programs you rely on are statically linked (they're probably not), then the `noexec` option is of little use in Linux. In addition, an attacker who has already gained root privileges will not be significantly hampered by filesystems mounted with special options, since these can often be remounted with the `-o remount` option. But by using mount flags, you can easily limit the possible attacks available to a hostile user before he gains root privileges.

```
-r-s--x--x 1 root root 16336 Feb 13 2003 /usr/bin/passwd

# <b>find / \( -perm -4000 -o -perm -2000 \) -type f -exec ls -la {} \;</b>

# <b>find / \( -perm -4000 -o -perm -2000 \) \</b>

<b> -type f -exec file {} \; | grep -v ELF</b>

$ <b>file /bin/sh</b>


/bin/sh: Mach-O executable ppc

0 4 * * * find / \( -perm -4000 -o -perm -2000 \) -type f \

    > /var/log/sidlog.new &&

    diff /var/log/sidlog.new /var/log/sidlog &&

    mv /var/log/sidlog.new /var/log/sidlog
```

This example will also leave a current list of SUID and SGID files in */var/log/sidlog*.

```
# <b>find / -type d \( -perm -g+w -o -perm -o+w \) -exec ls -lad {} \;</b>

# <b>find / -type d \( -perm -g+w -o -perm -o+w \) \</b>

    <b>-not -perm -a+t -exec ls -lad {} \;</b>
```

If you're using a system that creates a unique group for each user (e.g., you create a user *andrew*, which in turn creates a group *andrew* as the primary group), you may want to modify the commands to not scan for group-writable directories. (Otherwise, you will get a lot of output that really isn't pertinent.) To do this, run the command without the `-perm -g+w` portion.

```
# User ACL

u:[user]:<mode>

# Group ACL

g:[group]:<mode>

# Other ACL

o:<mode>

$ <b>touch myfile</b>

$ <b>ls -l myfile</b>

-rw-rw-r-- 1 andrew andrew 0 Oct 13 15:57 myfile

$ <b>setfacl -m u::---,g::---,o:--- myfile</b>

$ <b>ls -l myfile</b>

---------- 1 andrew andrew 0 Oct 13 15:57 myfile

$ <b>touch foo</b>

$ <b>setfacl -m u:jlope:rwx,g:wine:rwx ,o:--- foo</b>

$ <b>getfacl foo</b>

# file: foo

# owner: andrew

# group: andrew

user::rw-

user:jlope:rwx
```

```
group::---

group:wine:rwx

mask::rwx

other::---

$ <b>setfacl -m m:r-- foo</b>

$ <b>getfacl foo</b>

# file: foo

# owner: andrew

# group: andrew

user::rw-

user:jlope:rwx #effective:r--

group::---

group:wine:rwx #effective:r--

mask::r--

other::---

$ <b>mkdir mydir</b>

$ <b>setfacl -m d:u:jlope:rwx mydir</b>

$ <b>getfacl mydir</b>

# file: mydir

# owner: andrew
# group: andrew

user::rwx

group::---

other::---

default:user::rwx

default:user:jlope:rwx

default:group::---

default:mask::rwx

default:other::---

$ <b>touch mydir/bar</b>

$ <b>getfacl mydir/bar</b>

# file: mydir/bar

# owner: andrew

# group: andrew

user::rw-

user:jlope:rwx #effective:rw-

group::---

mask::rw-

other::---
```

As you may have noticed from the previous examples, you can list ACLs by using the `getfacl` command. This command is pretty straightforward and has only a few options. The most useful is the `-R` option, which allows you to list ACLs recursively and works very much like `ls -R`.

```
# <b>chattr +a</b>


<span class="docEmphBoldItalic">filename</span>

# <b>chflags sappnd</b>

<span class="docEmphBoldItalic">filename</span>

# <b>touch /var/log/logfile</b>

# <b>echo "append-only not set" > /var/log/logfile</b>

# <b>chattr +a /var/log/logfile</b>

# <b>echo "append-only set" > /var/log/logfile</b>

bash: /var/log/logfile: Operation not permitted

# <b>echo "appending to file" >> /var/log/logfile</b>

# <b>cat /var/log/logfile</b>

append-only not set
```

appending to file

```
# <b>tar xvfj lcap-0.0.3.tar.bz2 && cd lcap-0.0.3 && make</b>
```

```
# <b>./lcap CAP_LINUX_IMMUTABLE</b>
```

```
# <b>./lcap CAP_SYS_RAWIO</b>
```

```
kern.securelevel=1
```

Before doing this, you should be aware that adding append-only flags to your log files will most likely cause log rotation scripts to fail. However, doing this will greatly enhance the security of your audit trail, which will prove invaluable in the event of an incident.

Hack 6 Delegate Administrative Roles

HACK #6

Let others do your work for you without giving away root privileges.

The *sudo* utility can help you delegate some system responsibilities to other people, without giving away full root access. It is a setuid root binary that executes commands on an authorized user's behalf, after she has entered her current password.

As root, run */usr/sbin/visudo* to edit the list of users who can call *sudo*. The default *sudo* list looks something like this:

```
root ALL=(ALL) ALL
```

Unfortunately, many system administrators tend to use this entry as a template and grant unrestricted root access to all other admins unilaterally:

```
root ALL=(ALL) ALL

rob ALL=(ALL) ALL

jim ALL=(ALL) ALL

david ALL=(ALL) ALL
```

While this may allow you to give out root access without giving away the root password, this method is truly useful only when all of the *sudo* users can be completely trusted. When properly configured, the *sudo* utility provides tremendous flexibility for granting access to any number of commands, run as any arbitrary uid.

The syntax of the *sudo* line is:

```
user machine=(effective user) command
```

The first column specifies the *sudo* user. The next column defines the hosts in which this *sudo* entry is valid. This allows you to easily use a single *sudo* configuration across multiple machines.

For example, suppose you have a developer who needs root access on a development machine, but not on any other server:

```
peter beta.oreillynet.com=(ALL) ALL
```

The next column (in parentheses) specifies the effective user that may run the commands. This is very handy for allowing users to execute code as users other than root:

```
peter lists.oreillynet.com=(mailman) ALL
```

Finally, the last column specifies all of the commands that this user may run:

```
david ns.oreillynet.com=(bind) /usr/sbin/rndc,/usr/sbin/named
```

If you find yourself specifying large lists of commands (or, for that matter, users or machines), then take advantage of *sudo*'s Alias syntax. An Alias can be used in place of its respective entry on any line of the *sudo* configuration:

```
User_Alias ADMINS=rob,jim,david

User_Alias WEBMASTERS=peter,nancy

Runas_Alias DAEMONS=bind,www,smmsp,ircd

Host_Alias WEBSERVERS=www.oreillynet.com,www.oreilly.com,www.perl.com
```

```
Cmnd_Alias PROCS=/bin/kill,/bin/killall,/usr/bin/skill,/usr/bin/top

Cmnd_Alias APACHE=/usr/local/apache/bin/apachectl

WEBMASTERS WEBSERVERS=(www) APACHE

ADMINS ALL=(DAEMONS) ALL
```

It is also possible to specify system groups in place of the user specification, to allow any user who belongs to that group to execute commands. Just preface the group with a `%`, like this:

```
%wwwadmin WEBSERVERS=(www) APACHE
```

Now any user who is part of the wwwadmin group can execute *apachectl* as the www user on any of the web server machines.

One very useful feature is the `NOPASSWD`: flag. When present, the user won't have to enter a password before executing the command:

```
rob ALL=(ALL) NOPASSWD: PROCS
```

This will allow the user rob to execute *kill*, *killall*, *skill*, and *top* on any machine, as any user, without entering a password.

Finally, *sudo* can be a handy alternative to *su* for running commands at startup out of the system *rc* files:

```
(cd /usr/local/mysql; sudo -u mysql ./bin/safe_mysqld &)

sudo -u www /usr/local/apache/bin/apachectl start
```

For that to work at boot time, the default line `root ALL=(ALL) ALL` must be present.

Use *sudo* with the usual caveats that apply to setuid binaries. Particularly if you allow *sudo* to execute interactive commands (like editors) or any sort of compiler or interpreter, you should assume that it is possible that the *sudo* user will be able to execute arbitrary commands as the effective user. Still, under most circumstances this isn't a problem, and it's certainly preferable to giving away undue access to root privileges.

Rob Flickenger

```
# <b>gpg -import KEYS</b> # <b>gpg -verify apache_1.3.28.tar.gz.asc apache_1.3.28.tar.gz</b> gpg: Signature made Wed Jul 16 13:42:54 2003 PDT using DSA key ID 08C975E5

gpg: Good signature from "Jim Jagielski <jim@zend.com>"

gpg: aka "Jim Jagielski <jim@apache.org>"

gpg: aka "Jim Jagielski <jim@jaguNET.com>"

gpg: WARNING: This key is not certified with a trusted signature!

gpg: There is no indication that the signature belongs to the owner.

Fingerprint: 8B39 757B 1D8A 994D F243 3ED5 8B3A 601F 08C9 75E5

gpg: Signature made Wed Jul 16 13:42:54 2003 PDT using DSA key ID 08C975E5

gpg: Can't check signature: public key not found
```

```
#!/bin/sh

VENDOR_KEYRING=vendors.gpg

KEYSERVER=search.keyserver.net

KEYID="0x`gpg --verify $1 $2 2>&1 | grep 'key ID' | awk '{print $NF}'`"

gpg --no-default-keyring --keyring $VENDOR_KEYRING --recv-key \ --keyserver $KEYSERVER $KEYID

gpg --keyring $VENDOR_KEYRING --verify $1 $2
```

```
# <b>checksig apache_1.3.28.tar.gz.asc apache_1.3.28.tar.gz</b> gpg: requesting key 08C975E5 from HKP keyserver search.keyserver.net gpg: key 08C975E5: public key imported gpg: Total number processed: 1
```

```
gpg: imported: 1

gpg: Warning: using insecure memory!

gpg: please see http://www.gnupg.org/faq.html for more information gpg:
Signature made Wed Jul 16 13:42:54 2003 PDT using DSA key ID
08C975E5

gpg: Good signature from "Jim Jagielski <jim@zend.com>"

gpg: aka "Jim Jagielski <jim@apache.org>"

gpg: aka "Jim Jagielski <jim@jaguNET.com>"

gpg: checking the trustdb

gpg: no ultimately trusted keys found gpg: WARNING: This key is not
certified with a trusted signature!

gpg: There is no indication that the signature belongs to the owner.

Fingerprint: 8B39 757B 1D8A 994D F243 3ED5 8B3A 601F 08C9 75E5
```

With this small and quick script, both the number steps needed to verify a source package and the amount of time needed have been reduced. As with any good shell script, it should help you to be lazy in a good way: by doing more work properly, but with less effort on your part.

```
# <b>netstat -luntp</b>

Active Internet connections (only servers) Proto Recv-Q Send-Q Local
Address Foreign Address State PID/Program name tcp 0 0 0.0.0.0:22
0.0.0.0:* LISTEN 1679/sshd udp 0 0 0.0.0.0:68 0.0.0.0:* 1766/dhclient

# <b>netstat -a -n | egrep 'Proto|LISTEN'</b> Proto Recv-Q Send-Q Local
Address Foreign Address (state) tcp4 0 0 *.587 *.* LISTEN

tcp4 0 0 *.25 *.* LISTEN

tcp4 0 0 *.22 *.* LISTEN

tcp4 0 0 *.993 *.* LISTEN

tcp4 0 0 *.143 *.* LISTEN

tcp4 0 0 *.53 *.* LISTEN

# <b>grep -w 993 /etc/services</b> imaps 993/udp # imap4 protocol over
TLS/SSL

imaps 993/tcp # imap4 protocol over TLS/SSL

# <b>sockstat -4 -l</b>

USER COMMAND PID FD PROTO LOCAL ADDRESS FOREIGN
ADDRESS

root sendmail 1141 4 tcp4 *:25 *:*

root sendmail 1141 5 tcp4 *:587 *:*

root sshd 1138 3 tcp4 *:22 *:*

root inetd 1133 4 tcp4 *:143 *:*

root inetd 1133 5 tcp4 *:993 *:*
```

```
named named 1127 20 tcp4 *:53 *:*
```

```
named named 1127 21 udp4 *:53 *:*
```

```
named named 1127 22 udp4 *:1351 *:*
```

```
# <b>lsof -i -n | egrep 'COMMAND|LISTEN'</b> COMMAND PID USER
FD TYPE DEVICE SIZE/OFF NODE NAME
```

```
named 1127 named 20u IPv4 0xeb401dc0 0t0 TCP *:domain (LISTEN)
inetd 1133 root 4u IPv4 0xeb401ba0 0t0 TCP *:imap (LISTEN) inetd 1133
root 5u IPv4 0xeb401980 0t0 TCP *:imaps (LISTEN) sshd 1138 root 3u
IPv4 0xeb401760 0t0 TCP *:ssh (LISTEN) sendmail 1141 root 4u IPv4
0xeb41b7e0 0t0 TCP *:smtp (LISTEN) sendmail 1141 root 5u IPv4
0xeb438fa0 0t0 TCP *:submission (LISTEN)
```

Again, you can change the argument to `egrep` to display UDP sockets. However, this time use `UDP` instead of `udp4`, which makes the argument '`COMMAND|LISTEN|UDP`'. As mentioned earlier, not all UDP sockets will necessarily be associated with a daemon process.

```
Listen 192.168.0.23:80

<VirtualHost 192.168.0.23>

...



</VirtualHost>

[mysqld]



...



skip-networking

...

command=/usr/X11R6/bin/X

:0 local /usr/X11R6/bin/X -nolisten tcp

$ <b>startx -- -nolisten tcp</b>
```

Once you start X, fire up a terminal and see what is listening using lsof or netstat [Hack #8]. You should no longer see anything bound to port 6000.

```
# <b>mkdir -p /chroot_test/bin</b> # <b>cp /bin/bash /chroot_test/bin/</b>
# <b>chroot /chroot_test /bin/bash</b> chroot: /bin/bash: No such file or
directory
```

```
# <b>ldd /bin/bash</b>
```

```
libtermcap.so.2 => /lib/libtermcap.so.2 (0x4001a000) libdl.so.2 =>
/lib/libdl.so.2 (0x4001e000) libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000) # <b>mkdir -p
chroot_test/lib/tls && \</b> > <b>(cd /lib; \</b>
```

```
> <b>cp libtermcap.so.2 libdl.so.2 ld-linux.so.2 /chroot_test/lib; \</b> >
<b>cd tls; cp libc.so.6 /chroot_test/lib/tls)</b> # <b>chroot /chroot_test
/bin/bash</b> bash-2.05b#
```

```
bash-2.05b# <b>echo /*</b> /bin /lib
```

<b>jail</b>

<span class="docEmphBoldItalic">new root hostname ipaddr
command</span>

```
# <b>mkdir -p /jail_test/bin</b> # <b>cp /bin/sh /jail_test/sh</b> # <b>jail
/jail_test jail_test 192.168.0.40 /bin/sh</b> # <b>echo /*</b>
```

```
/bin
```

```
# <b>mkdir /jail_test</b> # <b>cd /usr/src</b>
```

```
# <b>make world DESTDIR=/jail_test</b> # <b>cd /etc && make
distribution DESTDIR=/jail_test -DNO_MAKEDEV_RUN</b> # <b>cd
/jail_test/dev && sh MAKEDEV jail</b> # <b>cd /jail_test && ln -s
dev/null kernel</b>
```

However, if you're planning to run just one service from within the jail, this is definitely overkill. Note that in the real world you'll most likely need to create */dev/null* and */dev/log* device nodes in your sandbox environment for most daemons to work correctly.

```
~$ <span class="docEmphBold">bzcat proftpd-1.2.6.tar.bz2 | tar xf -
</span>

~/proftpd-1.2.6/contrib$ <span class="docEmphBold">tar zvxf
../../mod_sql-4.08.tar.gz</span>

~/proftpd-1.2.6/contrib$ <span class="docEmphBold">cd .</span>.



~/proftpd-1.2.6$ ./configure --with-modules=mod_sql:mod_sql_mysql \

--with-includes=/usr/local/mysql/include/ \

--with-libraries=/usr/local/mysql/lib/

rob@catlin:~/proftpd-1.2.6$ <span class="docEmphBold">make && sudo
make install</span>

$ <span class="docEmphBold">mysqladmin create proftpd</span>

$ <span class="docEmphBold">mysql -e "grant select on proftpd.* to
proftpd@localhost \</span>

   <span class="docEmphBold">identified by 'secret';</span>"

CREATE TABLE users (



userid varchar(30) NOT NULL default '',

password varchar(30) NOT NULL default '',

uid int(11) default NULL,
gid int(11) default NULL,

homedir varchar(255) default NULL,

shell varchar(255) default NULL,

UNIQUE KEY uid (uid),

UNIQUE KEY userid (userid)

) TYPE=MyISAM;

CREATE TABLE groups (

groupname varchar(30) NOT NULL default '',
gid int(11) NOT NULL default '0',

members varchar(255) default NULL

) TYPE=MyISAM;
```

```
SQLConnectInfo proftpd proftpd secret
```

```
SQLAuthTypes crypt backend


SQLMinUserGID 111


SQLMinUserUID 111

SQLConnectInfo proftpd@dbhost:5678 somebody somepassword

mysql -e "insert into users values ('jimbo',PASSWORD('sHHH'),'111', \

   '111', '/usr/local/apache/htdocs','/bin/bash');" proftpd

# <span class="docEmphBold">proftpd -n -d 5</span>
```

Watch the messages as you attempt to connect, and you should be able to track down the source of difficulty. In my experience, it's almost always due to a failure to set something properly in *proftpd.conf*, usually regarding permissions.

The *mod_sql* module can do far more than I've shown here; it can connect to existing mysql databases with arbitrary table names, log all activity to the database, modify its user lookups with an arbitrary `WHERE` clause, and much more.

See Also

- The mod_sql home page at http://www.lastditcheffort.org/~aah/proftpd/mod_sql/
- The proftpd home page at http://www.proftpd.org/

Rob Flickenger (Linux Server Hacks)

Hack 12 Prevent Stack-Smashing Attacks

Learn how to prevent stack-based buffer overflows.

In C and C++, memory for local variables is allocated in a chunk of memory called the *stack*. Information pertaining to the control flow of a program is also maintained on the stack. If an array is allocated on the stack and that array is overrun (that is, more values are pushed into the array than the available space provides), an attacker can overwrite the control flow information that is also stored on the stack. This type of attack is often referred to as a *stack-smashing attack*.

Stack-smashing attacks are a serious problem, since an otherwise innocuous service (such as a web server or FTP server) can be made to execute arbitrary commands. Several technologies have been developed that attempt to protect programs against these attacks. Some are implemented in the compiler, such as IBM's ProPolice (http://www.trl.ibm.com/projects/security/ssp/) and the Stackguard (http://www.immunix.org/stackguard.html) versions of

GCC. Others are dynamic runtime solutions, such as LibSafe (http://www.research.avayalabs.com/project/libsafe/).

While recompiling the source gets to the heart of the buffer overflow attack, runtime solutions can protect programs when the source isn't available or recompiling simply

isn't feasible.

All of the compiler-based solutions work in much the same way, although there are some differences in the implementations. They work by placing a "canary" (which is

typically some random value) on the stack between the control flow information and the local variables. The code that is normally generated by the compiler to return from the function is modified to check the value of the canary on the stack; if it is not what it is supposed to be, the program is terminated immediately.

The idea behind using a canary is that an attacker attempting to mount a stack-smashing attack will have to overwrite the canary to overwrite the control flow information. By choosing a random value for the canary, the attacker cannot know what it is and thus cannot include it in the data used to

"smash" the stack.

When a program is distributed in source form, the developer of the program cannot enforce the use of StackGuard or ProPolice, because they are both nonstandard extensions to the GCC compiler. It is the responsibility of the person compiling the program to make use of one of these technologies.

For Linux systems, Avaya Labs's LibSafe technology is not

implemented as a compiler extension, but instead takes advantage of a feature of the dynamic loader that causes a dynamic library to be preloaded with every executable. Using LibSafe does not require the source code for the programs it protects, and it can be deployed on a system-wide basis.

LibSafe replaces the implementation of several standard functions that are known to be vulnerable to buffer overflows, such as `gets()`, `strcpy()`, and

`scanf()`. The replacement implementations attempt to compute the maximum possible size of a statically allocated buffer used as a destination buffer for writing, using a GCC built-in function that returns the address of the frame pointer. That address is normally the first piece of information on the stack following local variables. If an attempt is made to write more than the estimated size of the buffer, the program is terminated.

Unfortunately, there are several problems with the approach taken by LibSafe. First, it cannot accurately compute the size of a buffer; the best it can do is limit the size of the buffer to the difference between the start of the buffer and the frame pointer. Second, LibSafe's protections will not work with programs

that were compiled using the

`-fomit-frame-pointer` flag to GCC, an optimization that causes the compiler not to put a frame pointer on the stack. Although relatively useless, this is a popular optimization for programmers to employ. Finally, LibSafe will not work on SUID binaries without static linking or a similar trick.

In addition to providing protection against conventional stack-smashing attacks, the newest versions of LibSafe also provide some protection against format-string attacks. The format-string protection also requires access to the frame pointer because it attempts to filter out arguments that are not pointers into either the heap or the local variables on the stack.

In addition to user-space solutions, you can also opt to patch your kernel to use nonexecutable stacks and detect buffer overflow attacks. We'll do just that in **[Hack #13]** .

```
# <b>cd /usr/src/linux-2.4.24</b>

# <b>patch -p1 < ~andrew/grsecurity-1.9.13-2.4.24.patch</b>

# <b>find ./ -name \*.rej</b>

# <b>chpax -ps /usr/bin/java</b>

# <b>make dep clean && make bzImage</b>

# <b>make modules && make modules_install</b>
```

Then reboot with your new kernel. In addition to the kernel restrictions already in effect, you can now use `gradm` to set up ACLs for your system. We'll see how to do that in [Hack #14] .

As you can see, *grsecurity* is a complex but tremendously useful modification of the Linux kernel. For more detailed information on installing and configuring the patches, consult the extensive documentation at http://www.grsecurity.net/papers.php.

```
# <b>gradm -P</b>

Setting up grsecurity ACL password Password:

Re-enter Password:

Password written to /etc/grsec/pw.

# <b>/sbin/gradm -E</b>

# <b>/sbin/ifconfig eth0:1 192.168.0.59 up</b> SIOCSIFADDR:
Permission denied SIOCSIFFLAGS: Permission denied SIOCSIFFLAGS:
Permission denied

/sbin/ifconfig lo {

   / h

   /etc/grsec h -CAP_ALL

}

# <b>/sbin/ifconfig eth0:1 192.168.0.59 up</b> # <b>/sbin/ifconfig
eth0:1</b> eth0:1 Link encap:Ethernet HWaddr 00:0C:29:E2:2B:C1

   inet addr:192.168.0.59 Bcast:192.168.0.255 Mask:255.255.255.0

   UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

   Interrupt:10 Base address:0x10e0

# <b>gradm -a</b>

Password:

# <b>gradm -L -O stdout</b> /sbin/ifconfig o {
    /usr/share/locale/locale.alias r /usr/lib/locale/locale-archive r
/usr/lib/gconv/gconv-modules.cache r /proc/net/unix r /proc/net/dev r
/proc/net r /lib/ld-2.3.2.so x /lib/i686/libc-2.3.2.so rx /etc/ld.so.cache r
/sbin/ifconfig x /etc/grsec h / h

    -CAP_ALL

    +CAP_NET_ADMIN

}
```

Now you can replace the learning ACL for */sbin/ifconfig* in */etc/grsec/acl* with this one, and `ifconfig` should work. You can then follow this process for each program that needs special permissions to function. Just make sure to try out anything you will want to do with those programs, to ensure that *grsecurity*'s learning mode will detect that it needs to perform a particular system call or open a specific file.

Using *grsecurity* to lock down applications can seem like tedious work at first, but it will ultimately create a system that gives each process only the permissions it needs to do its jobno more, no less. When you need to build a highly secured platform, *grsecurity* can provide very finely grained control over just about everything the system can possibly do.

```
# Policy for named that uses named user and chroots to /var/named

# This policy works for the default configuration of named.


Policy: /usr/sbin/named, Emulation: native

native-accept: permit

native-bind: sockaddr match "inet-*:53" then permit

native-chdir: filename eq "/" then permit

native-chdir: filename eq "/namedb" then permit

native-chroot: filename eq "/var/named" then permit

native-connect: sockaddr eq "/dev/log" then permit

native-fsread: filename eq "/" then permit

native-fsread: filename eq "/dev/arandom" then permit

native-fsread: filename eq "/etc/group" then permit

filename sub "<non-existent filename>" then deny[enoent]

native-bind: sockaddr match "inet-*:53" then permit log

native-setgid: gid eq "70" then permit
```

This very brief overview covers the vast majority of the rules you will see. For full details on the *systrace* grammar, read the *systrace* manpage. If you want some help with creating your policies, you can also use systrace's automated mode [Hack #16] .

The original article that this hack is based on is available online at http://www.onlamp.com/pub/a/bsd/2003/01/30/Big_Scary_Daemons.html.

Michael Lucas

Hack 16 Automated Systrace Policy Creation

HACK #16

Let Systrace's automated mode do your work for you.

In a true paranoid's ideal world, system administrators would read the source code for every application on their system and be able to build system-call access policies by hand, relying only on their intimate understanding of every feature of the application. Most system administrators don't have that sort of time, and would have better things to do with that time if they did.

Luckily, *systrace* includes a policy-generation tool that will generate a policy listing for every system call that an application makes. You can use this policy as a starting point to narrow down the access you will allow the application. We'll use this method to generate a policy for `inetd`.

Use the `-A` flag to `systrace`, and include the full path to the program you want to run:

```
# systrace -A /usr/sbin/inetd
```

To pass flags to `inetd`, add them at the end of the command line.

Then use the program for which you're developing a policy. This system has ident, daytime, and time services open, so run programs that require those services. Fire up an IRC client to trigger ident requests, and telnet to ports 13 and 37 to get time services. Once you have put `inetd` through its paces, shut it down. `inetd` has no control program, so you need to kill it by process ID.

Checking the process list will show two processes:

```
# ps -ax | grep inet

24421 ??  Ixs     0:00.00 /usr/sbin/inetd

12929 ??  Is      0:00.01 systrace -A /usr/sbin/inetd
```

Do not kill the `systrace` process (PID 12929 in this example)that process has all the records of the system calls that `inetd` has made. Just kill the `inetd` process (PID 24421), and the `systrace` process will exit normally.

Now check your home directory for a *.systrace* directory, which will contain *systrace*'s first stab at an `inetd` policy. Remember, policies are placed in files named after the full path to the program, replacing slashes with underscores.

Here's the output of `ls`:

```
# ls .systrace

usr_libexec_identd   usr_sbin_inetd
```

`systrace` created two policies, not one. In addition to the expected policy for */usr/sbin/inetd*, there's one for */usr/libexec/identd*. This is because `inetd` implements time services internally, while ident calls a separate program to service requests. When `inetd` spawned `identd`, `systrace` captured the `identd` system calls as well.

By reading the policy, you can improve your understanding of what the program actually does. Look up each system call the program uses, and see if you can restrict access further. You'll probably want to look for ways to further restrict the policies that are automatically generated. However, these policies make for a good starting point.

Applying a policy to a program is much like creating the *systrace* policy itself; just run the program as an argument to `systrace`, using the `-a` option:

```
# systrace -a /usr/sbin/inetd
```

If the program tries to perform system calls not listed in the policy, they will fail. This may cause the program to behave unpredictably. *Systrace* will log failed entries in */var/log/messages*.

To edit a policy, just add the desired statement to the end of the rule list, and it will be picked up. You could do this by hand, of course, but that's the hard way. *Systrace* includes a tool to let you edit policies in real time, as the system call is made. This is excellent for use in a network operations center environment, where the person responsible for watching the network monitor can also be assigned to watch for system calls and bring them to the attention of the appropriate personnel. You can specify which program you wish to monitor by using `systrace`'s `-p` flag. This is called *attaching* to the program.

For example, earlier we saw two processes containing `inetd`. One was the actual `inetd` process, and the other was the `systrace` process managing `inetd`. Attach to the `systrace` process, not the actual program (to use the previous example, this would be PID 12929), and give the full path to the managed program as an argument:

```
# systrace -p 12929 /usr/sbin/inetd
```

At first nothing will happen. When the program attempts to make an unauthorized system call, however, a GUI will pop up. You will have the options to allow the system call, deny the system call, always permit the call, or always deny it. The program will hang until you make a decision, however, so decide quickly.

Note that these changes will only take effect so long as the current process is running. If you restart the program, you must also restart the attached `systrace` monitor, and any changes you set in the monitor are gone. You must add those rules to the policy if you want them to be permanent.

The original article that this hack is based on is available online at http://www.onlamp.com/pub/a/bsd/2003/02/27/Big_Scary_Daemons.html.

Michael Lucas

```
#%PAM-1.0

auth required pam_securetty.so

auth required pam_stack.so service=system-auth

auth required pam_nologin.so

account required pam_stack.so service=system-auth

password required pam_stack.so service=system-auth

session required pam_stack.so service=system-auth

session optional pam_console.so

#%PAM-1.0

# This file is auto-generated.

# User changes will be destroyed the next time authconfig is run.

auth required /lib/security/$ISA/pam_env.so

auth sufficient /lib/security/$ISA/pam_unix.so likeauth nullok

auth required /lib/security/$ISA/pam_deny.so

account required /lib/security/$ISA/pam_unix.so

password required /lib/security/$ISA/pam_cracklib.so retry=3 type=

password sufficient /lib/security/$ISA/pam_unix.so nullok use_authtok md5
shadow

password required /lib/security/$ISA/pam_deny.so

session required /lib/security/$ISA/pam_limits.so
session required /lib/security/$ISA/pam_unix.so
```

```
account required pam_access.so
```

<span class="docEmphBoldItalic">permission</span>

:

<span class="docEmphBoldItalic">users</span>

:

<span class="docEmphBoldItalic">origins</span>

```
- : andrew : colossus
```

<tt><i>services</i></tt>;<tt><i>devices</i></tt>;<tt><i>users</i></tt>;
<tt><i>times</i></tt>

```
$ <b>ls -1 /etc/pam.d</b>
```

authconfig

chfn

chsh

halt

internet-druid

kbdrate

login

neat

other

passwd

poweroff

ppp

reboot

redhat-config-mouse

redhat-config-network

redhat-config-network-cmd

redhat-config-network-druid

rhn_register

setup

smtp

sshd

su

sudo

system-auth

up2date

up2date-config

up2date-nox

```
vlock

account required /lib/security/$ISA/pam_time.so

system-auth;!ttyp*;andrew;Wk1700-0800|Wd0000-2400

sshd;ttyp*;!andrew;Fr1900-0700
```

As you can see, there's a lot of flexibility for creating entries, thanks to the logical Boolean operators that are available. Just make sure that you remember to configure the service file in */etc/pam.d* for use with `pam_time` when you create entries in */etc/security/time.conf*.

```
$ <b>bash -r</b>

bash: SHELL: readonly variable

bash: PATH: readonly variable

bash-2.05b$ <b>ls</b> bash: ls: No such file or directory bash-2.05b$
<b>/bin/ls</b> bash: /sbin/ls: restricted: cannot specify `/' in command
names bash-2.05b$ <b>exit</b> $ <b>ln -s /bin/ls .</b> $ <b>bash -r</b>

bash-2.05b$ <b>ls -la</b> total 24

drwx------ 2 andrew andrew 4096 Oct 20 08:01 .

drwxr-xr-x 4 root root 4096 Oct 20 14:16 ..

-rw------- 1 andrew andrew 18 Oct 20 08:00 .bash_history -rw-r--r-- 1
andrew andrew 24 Oct 20 14:16 .bash_logout -rw-r--r-- 1 andrew andrew
197 Oct 20 07:59 .bash_profile -rw-r--r-- 1 andrew andrew 127 Oct 20
07:57 .bashrc lrwxrwxrwx 1 andrew andrew 7 Oct 20 08:01 ls -> /bin/ls

$ <b>rksh</b>

$ <b>ls -la</b>

total 24

drwx------ 2 andrew andrew 4096 Oct 20 08:01 .

drwxr-xr-x 4 root root 4096 Oct 20 14:16 ..

-rw------- 1 andrew andrew 18 Oct 20 08:00 .bash_history -rw-r--r-- 1
andrew andrew 24 Oct 20 14:16 .bash_logout -rw-r--r-- 1 andrew andrew
197 Oct 20 07:59 .bash_profile -rw-r--r-- 1 andrew andrew 127 Oct 20
07:57 .bashrc lrwxrwxrwx 1 andrew andrew 7 Oct 20 08:01 ls -> /bin/ls $
<b>which ls</b>

/bin/ls
```

```
$ <b>exit</b>

$ <b>export PATH=.</b> $ <b>/bin/rksh</b>

$ <b>/bin/ls</b>

/bin/rksh: /bin/ls: restricted

$ <b>exit</b>

$ <b>ln -s /bin/ls .</b> $ <b>ls -la</b>

total 24

drwx------ 2 andrew andrew 4096 Oct 20 08:01 .

drwxr-xr-x 4 root root 4096 Oct 20 14:16 ..

-rw------- 1 andrew andrew 18 Oct 20 08:00 .bash_history -rw-r--r-- 1
andrew andrew 24 Oct 20 14:16 .bash_logout -rw-r--r-- 1 andrew andrew
197 Oct 20 07:59 .bash_profile -rw-r--r-- 1 andrew andrew 127 Oct 20
07:57 .bashrc lrwxrwxrwx 1 andrew andrew 7 Oct 20 08:01 ls -> /bin/ls
```

Restricted shells are incredibly easy to set up and can provide minimal restricted access. They may not be able to keep out determined attackers, but they certainly make a hostile user's job much more difficult.

```
<tt><i>domain type resource value</i></tt>

guest soft nofile 1000

guest hard nofile 2000

# <b>su - guest</b> $ <b>ulimit -a</b> core file size (blocks, -c) 0

data seg size (kbytes, -d) unlimited file size (blocks, -f) unlimited max
locked memory (kbytes, -l) unlimited max memory size (kbytes, -m)
unlimited open files (-n) 1000

pipe size (512 bytes, -p) 8

stack size (kbytes, -s) 8192

cpu time (seconds, -t) unlimited max user processes (-u) 1024

virtual memory (kbytes, -v) unlimited $ <b>ulimit -n 2000</b> $ <b>ulimit
-n</b>
```

2000

```
$ <b>ulimit -n 2001</b> -bash: ulimit: open files: cannot modify limit: Operation not permitted
```

There you have it. In addition to open files, you can create resource limits for any number of other resources and apply them to specific users or entire groups. As you can see, `pam_limits` is quite powerful and useful in that it doesn't rely upon the shell for enforcement.

```
# <b>rpm -ivh autorpm-3.3-1.noarch.rpm</b>
```

Although a tarball is also available, installation is a little more tricky than the typical `make; make install`, and so it is recommended that you stick to installing from the RPM package.

By default, AutoRPM is configured to monitor for updated packages for Red Hat's Linux distribution. However, you can configure it to monitor any file repository of your choosing, such as one for SuSe or Mandrake.

Chapter 2. Windows Host Security

Hacks #21-30

This chapter shows you some ways to keep your Windows system up-to-date and secure, thereby making your network a safer place to work (and have fun). Although many may scoff at the mention of Windows and security in the same sentence, you actually can make a Windows system fairly secure without too much effort.

One of the main reasons that Windows gets a bad rap is the poorly administered state in which Windows machines seem to be kept. The recent deluge of worm and virus attacks that have brought down many a network shows this to hold true. A lot of this can be traced back to the "ease" of administration that

Windows seems to provide by effectively keeping the Windows administrator out of the loop about the inner workings of her environmenteffectively wresting control from the system administrator's hands.

This chapter seeks to remedy that to some degree by showing you ways to see exactly what your server is really doing. While this may seem old hat to a Unix sysadmin, getting details on open ports and running services is often a new concept to the average Windows administrator.

In addition, this chapter shows you how to disable some Windows "features," such as sharing out all

your files automatically and truncating log files.

You'll also learn how to enable some of the auditing and logging features of Windows, to give you early warning of possible security incidents (rather than waiting for the angry phone call from someone at the wrong end of a denial-of-service attack originating from your network).

```
C:\> Program Files\Microsoft Baseline Security Analyzer> <b>mbsacli
/hf</b> Microsoft Baseline Security Analyzer

Version 1.1.1

Powered by HFNetChk Technology - Version 3.82.0.1

Copyright (C) Shavlik Technologies, 2001-2003

Developed for Microsoft by Shavlik Technologies, LLC

info@shavlik.com (www.shavlik.com)

Please use the -v switch to view details for Patch NOT Found, Warning and
Note messages Attempting to get cab from http://go.microsoft.com/fwlink/?
LinkId=16932

XML successfully loaded.
```

Scanning PLUNDER

....

```
Done scanning PLUNDER

----------------------------

PLUNDER(192.168.0.65)

----------------------------

   * WINDOWS XP SP1

   Note MS02-008 317244

   Warning MS02-055 323255

   Note MS03-008 814078

   Note MS03-030 819696

   Patch NOT Found MS03-041 823182

   Patch NOT Found MS03-044 825119

   Patch NOT Found MS03-045 824141

   Patch NOT Found MS03-049 828035

   Note MS03-051 813360

   * INTERNET EXPLORER 6 SP1

   Patch NOT Found MS03-048 824145

   * WINDOWS MEDIA PLAYER FOR WINDOWS XP SP1

   Information

   All necessary hotfixes have been applied.
```

Scanning PLUNDER

....

```
Done scanning PLUNDER

----------------------------

PLUNDER(192.168.0.65)

----------------------------

   * WINDOWS XP SP1

   Note MS02-008 317244

   Please refer to Q306460 for a detailed explanation.

   Warning MS02-055 323255

   File C:\WINDOWS\system32\hhctrl.ocx has a file version [5.2.3735.0]
greater than what is expected [5.2.3669.0].

   Note MS03-008 814078

   Please refer to Q306460 for a detailed explanation.

   Note MS03-030 819696

   Please refer to Q306460 for a detailed explanation.

   Patch NOT Found MS03-041 823182

   File C:\WINDOWS\system32\cryptui.dll has a file version
[5.131.2600.1106] that is less than what is expected [5.131.2600.1243].

   Patch NOT Found MS03-044 825119

   File C:\WINDOWS\system32\itircl.dll has a file version [5.2.3644.0] that
is less than what is expected [5.2.3790.80].

   Patch NOT Found MS03-045 824141
```

File C:\WINDOWS\system32\user32.dll has a file version [5.1.2600.1134] that is less than what is expected [5.1.2600.1255].

Patch NOT Found MS03-049 828035

File C:\WINDOWS\system32\msgsvc.dll has a file version [5.1.2600.0] that is less than what is expected [5.1.2600.1309].

Note MS03-051 813360

Please refer to Q306460 for a detailed explanation.

* INTERNET EXPLORER 6 SP1

Patch NOT Found MS03-048 824145

The registry key **SOFTWARE\Microsoft\Internet Explorer\ActiveX

Compatibility\{69DEAF94-AF66-11D3-BEC0-00105AA9B6AE}** does not exist. It is required for this patch to be considered installed.

* WINDOWS MEDIA PLAYER FOR WINDOWS XP SP1

Information

All necessary hotfixes have been applied.

Scanning PLUNDER

....

```
Done scanning PLUNDER

----------------------------

PLUNDER(192.168.0.65)

----------------------------

   * WINDOWS XP SP1

   Information

   All necessary hotfixes have been applied.

   * INTERNET EXPLORER 6 SP1

   Information

   All necessary hotfixes have been applied.

   * WINDOWS MEDIA PLAYER FOR WINDOWS XP SP1

   Information

   All necessary hotfixes have been applied.
```

```
mbsacli /hf -h PLUNDER

mbsacli /hf -i 192.168.0.65

mbsacli /hf -r 192.168.1.123 - 192.168.1.172
```

All of these options are very flexible, and you can use them in any combination to specify which remote systems will be scanned.

In addition to specifying remote systems by NetBIOS name and IP address, you can also scan systems by domain name by using the `-d` option, or you can scan your entire local network segment by using the `-n` command-line option.

When scanning systems from a personal workstation, the `-u` and `-p` options can prove useful. These allow you to specify a username and password to use when accessing the remote systems. These switches are particularly handy if you don't normally log in using the Administrator account. The account that is specified with the `-u` option will of course need to have Administrator privileges on the remote machines being scanned.

Also, if you're scanning a large number of systems, you might want to use the `-t` option. This allows you to specify the number of threads used by the scanner, and increasing this value generally will speed up scanning. Valid values are from 1 to 128; the default value is 64.

If you are scanning more than one machine, a huge amount of data will simply be dumped to the screen. Use the `-f` option to specify a file to store the results of the scan in, and view it at your leisure using a text editor.

HFNetChk is a very flexible tool and can be used to check the update status of a large number of machines in a very short amount of time. It is especially useful when a new worm has come onto the scene and you need to know if all of your systems are up-to-date on their patches.

See Also

- Frequently Asked Questions about the Microsoft Network Security Hotfix Checker (*Hfnetchk.exe*) Tool: Knowledge Base Article 305385, at http://support.microsoft.com/default.aspx?scid=kb;EN-US;

```
C:\> <b>handle</b>

<span class="docEmphBoldItalic">filename</span>

C:\> <b>handle -p iexplore</b>

Handle v2.10

Copyright (C) 1997-2003 Mark Russinovich

Sysinternals - www.sysinternals.com

------------------------------------------------------------------------------

IEXPLORE.EXE pid: 688 PLUNDER\andrew

   98: Section
\BaseNamedObjects\MTXCOMM_MEMORY_MAPPED_FILE

   9c: Section \BaseNamedObjects\MtxWndList 12c: Section
\BaseNamedObjects\__R_0000000000d4_SMem_ _

   18c: File C:\Documents and Settings\andrew\Local Settings\Temporary
Internet Files\Content.IE5\index.dat

   198: Section \BaseNamedObjects\C:_Documents and
Settings_andrew_Local Settings_Temporary Internet
Files_Content.IE5_index.dat_3194880

   1a0: File C:\Documents and Settings\andrew\Cookies\index.dat 1a8: File
C:\Documents and Settings\andrew\Local Settings\History\History.IE5\
index.dat

   1ac: Section \BaseNamedObjects\C:_Documents and
Settings_andrew_Local Settings_History_History.IE5_index.dat_245760

   1b8: Section \BaseNamedObjects\C:_Documents and
Settings_andrew_Cookies_index.dat_81920
```

```
   228: Section \BaseNamedObjects\UrlZonesSM_andrew 2a4: Section
\BaseNamedObjects\SENS Information Cache 540: File C:\Documents and
Settings\andrew\Application Data\Microsoft\SystemCertificates\My

   574: File C:\Documents and Settings\All Users\Desktop 5b4: Section
\BaseNamedObjects\mmGlobalPnpInfo 5cc: File
C:\WINNT\system32\mshtml.tlb 614: Section
\BaseNamedObjects\WDMAUD_Callbacks 640: File
C:\WINNT\system32\Macromed\Flash\Flash.ocx 648: File
C:\WINNT\system32\STDOLE2.TLB

   6a4: File \Dfs

   6b4: File C:\Documents and Settings\andrew\Desktop 6c8: File
C:\Documents and Settings\andrew\Local Settings\ Temporary Internet
Files\Content.IE5\Q5USFST0\softwareDownloadIndex[1].htm 70c: Section
\BaseNamedObjects\MSIMGSIZECacheMap 758: File
C:\WINNT\system32\iepeers.dll 75c: File C:\Documents and
Settings\andrew\Desktop 770: Section \BaseNamedObjects\RotHintTable

C:\> <b>handle -p iexplore handle</b>

Handle v2.10

Copyright (C) 1997-2003 Mark Russinovich

Sysinternals - www.sysinternals.com

IEXPLORE.EXE pid: 1396 C:\Documents and Settings\andrew\Local
Settings\Temporary Internet Files\Content.IE5\H1EZGFSH\handle[1].htm
```

Additionally, if you wanted to list all types of resources, you could use the -a option. Handle is quite a powerful tool, and any of its command-line options can be mixed together to quickly narrow your search and find just what you want.

```
C:\> <b>fport /p</b>

FPort v2.0 - TCP/IP Process to Port Mapper Copyright 2000 by
Foundstone, Inc.

http://www.foundstone.com
```

```
Pid Process Port Proto Path

432 svchost -> 135 TCP C:\WINNT\system32\svchost.exe 8 System -> 139
TCP

8 System -> 445 TCP

672 MSTask -> 1025 TCP C:\WINNT\system32\MSTask.exe 8 System ->
1028 TCP

8 System -> 1031 TCP

1116 navapw32 -> 1035 TCP C:\PROGRA~1\NORTON~1\navapw32.exe
788 svchost -> 1551 TCP C:\WINNT\system32\svchost.exe 788 svchost ->
1553 TCP C:\WINNT\system32\svchost.exe 788 svchost -> 1558 TCP
C:\WINNT\system32\svchost.exe 1328 svchost -> 1565 TCP
C:\WINNT\System32\svchost.exe 8 System -> 1860 TCP

1580 putty -> 3134 TCP C:\WINNT\putty.exe 772 WinVNC -> 5800 TCP
C:\Program Files\TightVNC\WinVNC.exe 772 WinVNC -> 5900 TCP
C:\Program Files\TightVNC\WinVNC.exe 432 svchost -> 135 UDP
C:\WINNT\system32\svchost.exe 8 System -> 137 UDP

8 System -> 138 UDP

8 System -> 445 UDP

256 lsass -> 500 UDP C:\WINNT\system32\lsass.exe 244 services -> 1027
UDP C:\WINNT\system32\services.exe 688 IEXPLORE -> 2204 UDP
C:\Program Files\Internet Explorer\IEXPLORE.EXE

1396 IEXPLORE -> 3104 UDP C:\Program Files\Internet
Explorer\IEXPLORE.EXE

256 lsass -> 4500 UDP C:\WINNT\system32\lsass.exe
```

Notice that there are some processes listedsuch as `navapw32`, `putty`, and `IEXPLORE`that don't appear to be services. These show up in the output because FPort lists all open ports, not just opened ports that are listening.

While FPort is not as powerful as some of the commands available under other operating systems, it is still a valuable, quick, and easy-to-use tool that is a great addition to Windows.

Hack 24 Enable Auditing

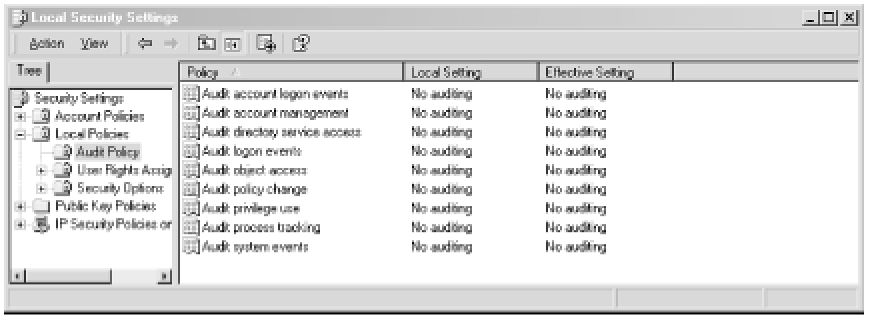
Log suspicious activity to help spot intrusions.

Windows 2000 includes some very powerful auditing features, but unfortunately they are all disabled by default. Windows 2003 has corrected this by enabling some features by default, but it is still wise to check that you are tracking precisely what you want to audit. Using these capabilities, you can monitor failed logins, account management events, file access, privilege use, and more. You can also log

security policy changes as well as system events.

To enable auditing in any one of these areas, locate and double-click the Administrative Tools icon in the Control Panel. Now find and double-click the Local Security Policy icon. Expand the Local Policies tree node, and you should see something similar to Figure 2-1.

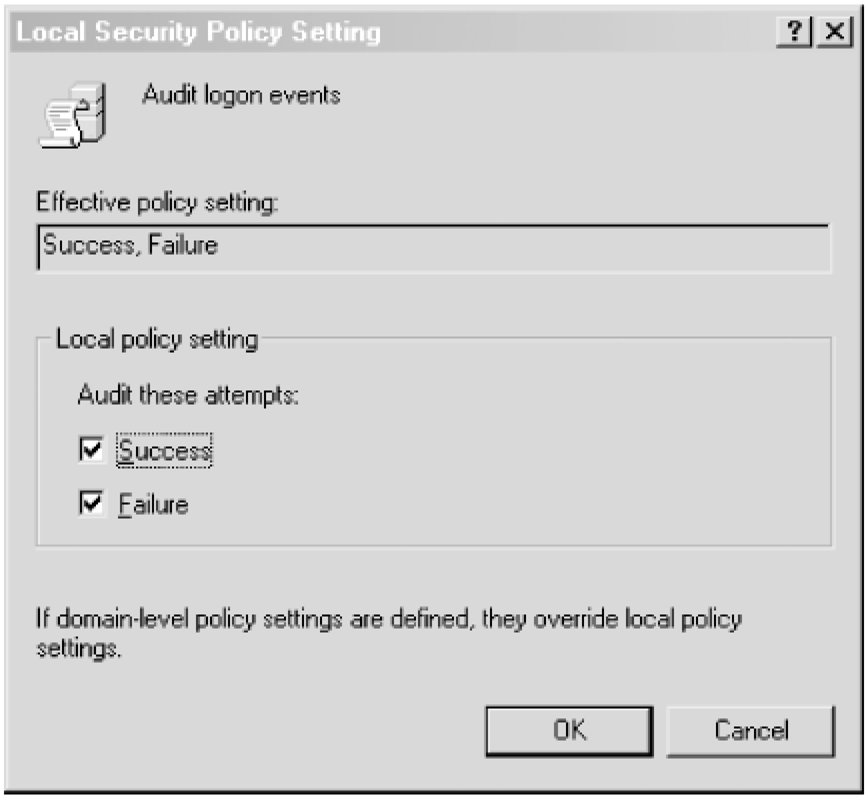
Figure 2-1. Audit Policy settings in the Local Security Settings applet



Policy	Local Setting	Effective Setting
Audit account logon events	No auditing	No auditing
Audit account management	No auditing	No auditing
Audit directory service access	No auditing	No auditing
Audit logon events	No auditing	No auditing
Audit object access	No auditing	No auditing
Audit policy change	No auditing	No auditing
Audit privilege use	No auditing	No auditing
Audit process tracking	No auditing	No auditing
Audit system events	No auditing	No auditing

Now you can go through each of the audit policies and check whether to log successes or failures for each type. You can do this by double-clicking the policy you wish to modify, located in the right pane of the window. After double-clicking, you should see a dialog similar to Figure 2-2.

Figure 2-2. The "Audit logon events" dialog


Local Security Policy Setting

Audit logon events

Effective policy setting:

Success, Failure

Local policy setting

Audit these attempts:

[✓] Success

[✓] Failure

If domain-level policy settings are defined, they override local policy settings.

Leaving auditing off is akin to not logging anything at all, so you should enable auditing for all policies. Once you've

enabled auditing for a particular policy, you should begin to see entries in the event logs for when a particular audit event occurs.

For example, once you have enabled logon event auditing, you should begin to see entries for logon successes and failures in the system's security event log.

Hack 25 Secure Your Event Logs

Keep your system's logs from being tampered with.

Windows has some very powerful logging features. Unfortunately, by default the event logs are not protected against unauthorized access or modification. You may not realize that even though you have to view the logs through the Event Viewer, the event logs are simply regular files just like any other. To secure them, all we have to do is locate them and apply the proper ACLs.

Unless their location has been changed through the registry, you should be able to find the logs in the

%SystemRoot%\system32\config directory.

The three files that correspond to the Application Log, Security Log, and System Log are *AppEvent.Evt*,

SecEvent.Evt, and

SysEvent.Evt, respectively. Now, apply ACLs to limit access to only Administrator accounts. You can do this by bringing up the Properties dialog for the files and clicking the Security tab. After you've done this, remove any users or groups other than Administrators and SYSTEM from the top pane.

Hack 26 Change Your Maximum Log File Sizes

Change your log properties so that they see the whole picture.

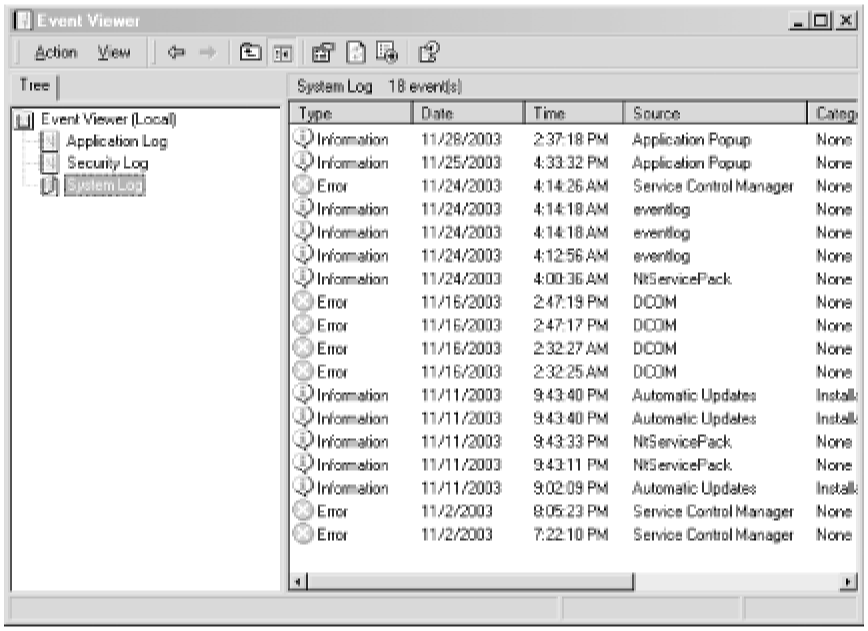
From a security point of view, logs are one of the most important assets contained on a server. After all, without logs how will you know if or when someone has gained access to your machine? Therefore, it is imperative that your logs not miss a beat. If you're trying to track down the

source of an incident, having missing log entries is not much better than having no logs at all.

One common problem is that the maximum log size is set too lowthe default is a measly 512KB. To change this, open the Administrative Tools control panel, and then open the Event Viewer.

You should now see something similar to Figure 2-3.

Figure 2-3. The Windows Event Viewer

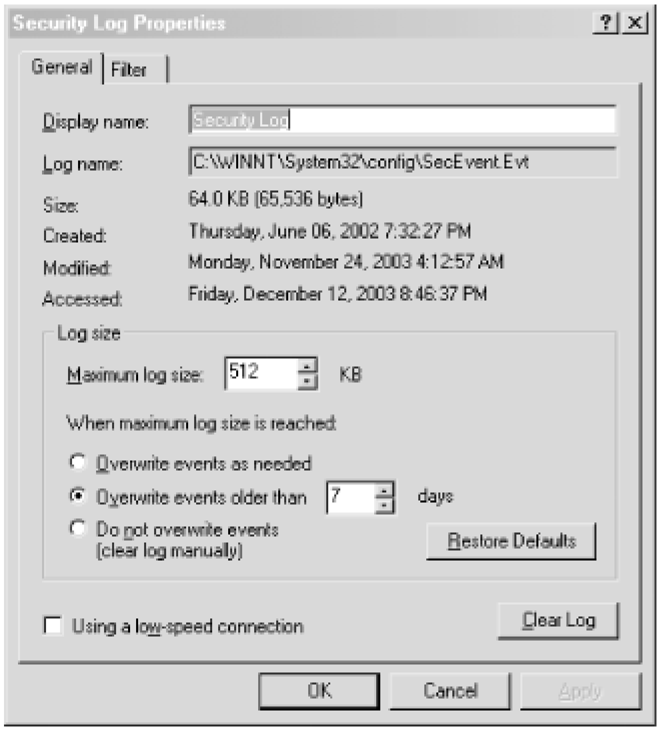


System Log 18 event(s)

Type	Date	Time	Source	Categ
Information	11/28/2003	2:37:18 PM	Application Popup	None
Information	11/25/2003	4:33:32 PM	Application Popup	None
Error	11/24/2003	4:14:26 AM	Service Control Manager	None
Information	11/24/2003	4:14:18 AM	eventlog	None
Information	11/24/2003	4:14:18 AM	eventlog	None
Information	11/24/2003	4:12:56 AM	eventlog	None
Information	11/24/2003	4:00:36 AM	NtServicePack	None
Error	11/16/2003	2:47:19 PM	DCOM	None
Error	11/16/2003	2:47:17 PM	DCOM	None
Error	11/16/2003	2:32:27 AM	DCOM	None
Error	11/16/2003	2:32:25 AM	DCOM	None
Information	11/11/2003	9:43:40 PM	Automatic Updates	Install
Information	11/11/2003	9:43:40 PM	Automatic Updates	Install
Information	11/11/2003	9:43:33 PM	NtServicePack	None
Information	11/11/2003	9:43:11 PM	NtServicePack	None
Information	11/11/2003	9:02:09 PM	Automatic Updates	Install
Error	11/2/2003	8:05:23 PM	Service Control Manager	None
Error	11/2/2003	7:22:10 PM	Service Control Manager	None

After you have done this, select one of the log files from the left pane of the Event Viewer window and right-click it. Now select the Properties menu item. You should now see something similar to Figure 2-4.

Figure 2-4. Security Log Properties



Display name:	Security Log
Log name:	C:\WINNT\System32\config\SecEvent.Evt
Size:	64.0 KB (65,536 bytes)
Created:	Thursday, June 06, 2002 7:32:27 PM
Modified:	Monday, November 24, 2003 4:12:57 AM
Accessed:	Friday, December 12, 2003 8:46:37 PM

Log size

Maximum log size: 512 KB

When maximum log size is reached:

- ( ) Overwrite events as needed
- [✓ Overwrite events older than 7 days]
- ( ) Do not overwrite events (clear log manually)

[ ] Using a low-speed connection

Now locate the text input box with the label "Maximum log size". You can type in

the new maximum size directly, or you can use the arrows next to the text box to change the value. Anything above 1MB is good to use here.

It all depends on how often you want to review and archive your logs.

However, keep in mind that having very large log files won't inherently slow down the machine, but can slow

down the Event Viewer when you're trying to view the

logs. While you're here, you may also want to change

the behavior for when the log file reaches its maximum size. By default, it will start overwriting log entries that are older than seven days with newer log entries. It is recommended that you change this value to something highersay 31 days. Alternatively, you could elect not to have logs overwritten automatically at all, in which case you'll need to clear the log manually.

```
C:\><b>net share</b>
```

```
Share name Resource Remark

---------------------------------------------------------------------------



IPC$ Remote IPC The command completed successfully.
```

Before doing this, you should be sure that disabling these shares will not negatively affect your environment. Lack of these shares can cause some system management softwaresuch as HFNetChk [Hack #21] or System Management Serverto not work. This is because software like this depends on remote access to the default administrative shares in order to access the contents of the systems disks.

Hack 28 Encrypt Your Temp Folder

Keep prying eyes out of your temporary files.

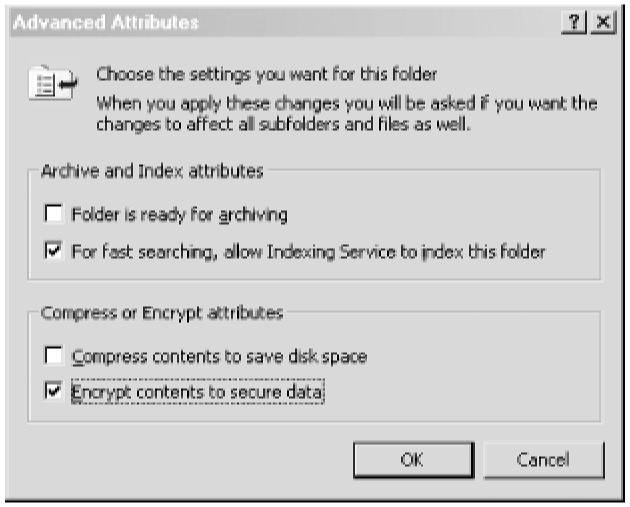
Many Windows applications will create intermediary files while they do their work. They typically store these files in a temporary folder within the current user's settings directory. Most often these files

are created world-readable and aren't always cleaned up when the program exits. How would you like it if your word processor left a copy of the last document you were working on for anyone to come across and read? Not a pretty thought, is it?

One way to guard against this situation is to encrypt your temporary files folder. To do this, open an Explorer window and go to the *C:\Documents and Settings\<username>\Local Settings* folder. In this folder you should see another folder called *Temp*. This is the folder that

holds the temporary files. Right-click the folder and bring up its Properties dialog. Make sure the General tab is selected, and click the button labeled Advanced. This will bring up an Advanced Attributes dialog, as seen in Figure 2-6. Here you can choose to encrypt the folder.

Figure 2-6. The Temp folder's Advanced Attributes dialog


Advanced Attributes

Choose the settings you want for this folder

When you apply these changes you will be asked if you want the changes to affect all subfolders and files as well.

Archive and Index attributes

- [ ] Folder is ready for archiving
- [✓] For fast searching, allow Indexing Service to index this folder

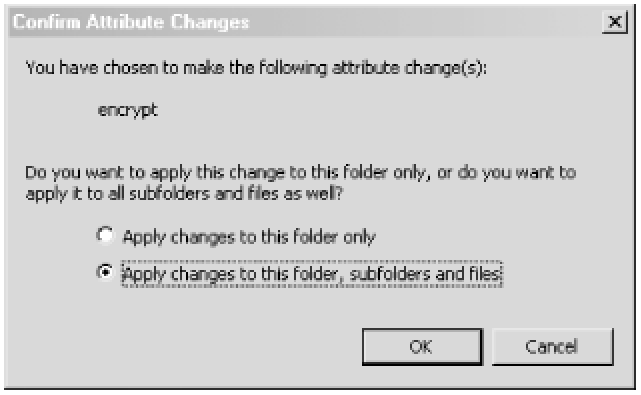
Compress or Encrypt attributes

- [ ] Compress contents to save disk space
- [✓] Encrypt contents to secure data

Check the "Encrypt contents to secure data" box and click the OK button. When you have

done that, click the Apply button in the Properties dialog. Another dialog (as seen in Figure 2-7) will open asking you whether you would like the encryption to apply recursively.

Figure 2-7. Confirm the choice of encryption and make it recursive


Confirm Attribute Changes

You have chosen to make the following attribute change(s):

encrypt

Do you want to apply this change to this folder only, or do you want to apply it to all subfolders and files as well?

- ○ Apply changes to this folder only
- ● Apply changes to this folder, subfolders and files

To apply the encryption recursively, choose the "Apply changes to this folder, subfolders and

files" option. This will

automatically create a public-key pair if you have never encrypted any files before. Otherwise, Windows will use the public key that it generated for you previously. When decrypting, Windows ensures that the private keys are stored in nonpaged kernel memory, so that the decryption key will never be left in the paging file. Unfortunately, the encryption algorithm used, DESX, is barely an improvement on DES

and is nowhere near as strong as 3DES. However, it serves the purpose of transparently encrypting temporary files very well. If you want to encrypt other files, it is suggested you use a third-party utility such as GnuPG (http://www.gnupg.org), which has Windows binaries available on its web site.

Hack 29 Clear the Paging File at Shutdown

HACK #29

Prevent information leaks by automatically clearing the swap file before shutting down.

Virtual memory management (VMM) is truly a wonderful thing. It protects programs from one another and lets them think that they have more memory available than is physically in the system. To accomplish this, the VMM uses what is called a *paging file*

.

As you run more and more programs over the course of time, you'll begin to run out of physical memory. Since

things can start to go awry when this happens, the memory manager will look for the least frequently used pieces of memory owned by programs that aren't actively doing anything at the

moment and write the chunks of memory out to the disk (i.e., the virtual memory). This is known as

swapping.

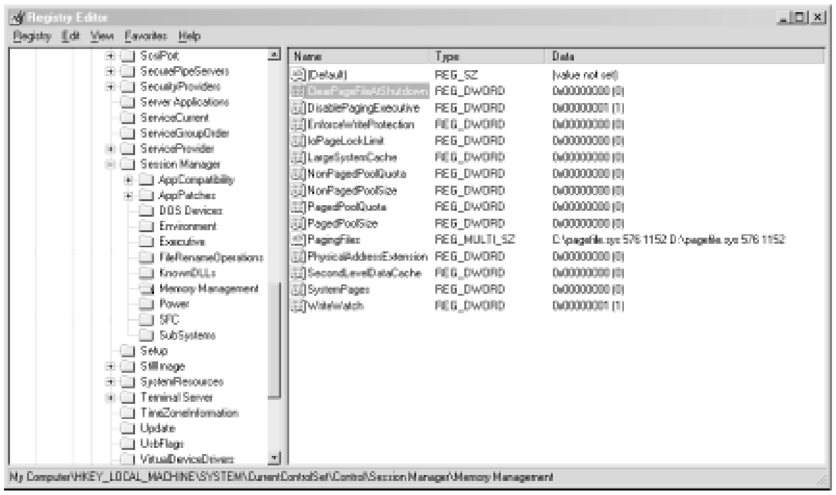
However, there is one possibly bad side effect of this feature: if a program containing confidential information in its memory space is running, the memory containing such information may be written out to disk. This is fine when the operating system is running and there are safeguards to prevent the paging file from being read, but what about when the system is off or booted into a different operating system?

This is where this hack comes in handy. What we're going to do is tell the operating system to overwrite the paging file with zeros when it shuts down. Keep in mind that this will not work if the cord is pulled from the system or the system is shut down improperly, since this overwrite will only be done during a proper shutdown.

To enable this feature of Windows, we must edit the system registry. To do this, open the Registry and find the

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management key. You should now see something that looks like Figure 2-8.

Figure 2-8. The Memory Management registry key



```
SciPort
SecurePipeServers
SecurityProviders
Server Applications
ServiceCurrent
ServiceGroupOrder
ServiceProvider
Session Manager
    AppCompatibility
    AppPatches
    DOS Devices
    Environment
    Executive
    FileRenameOperations
    KnownDLLs
    Memory Management
    Power
    SFC
    SubSystems
Setup
StillImage
SystemResources
Terminal Server
TimeZoneInformation
Update
UsbFlags
VirtualDeviceDrivers
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
ClearPageFileAtShutdown	REG_DWORD	0x00000000 (0)
DisablePagingExecutive	REG_DWORD	0x00000001 (1)
EnforceWriteProtection	REG_DWORD	0x00000000 (0)
IoPageLockLimit	REG_DWORD	0x00000000 (0)
LargeSystemCache	REG_DWORD	0x00000000 (0)
NonPagedPoolQuota	REG_DWORD	0x00000000 (0)
NonPagedPoolSize	REG_DWORD	0x00000000 (0)
PagedPoolQuota	REG_DWORD	0x00000000 (0)
PagedPoolSize	REG_DWORD	0x00000000 (0)
PagingFiles	REG_MULTI_SZ	C:\pagefile.sys 576 1152 D:\pagefile.sys 576 1152
PhysicalAddressExtension	REG_DWORD	0x00000000 (0)
SecondLevelDataCache	REG_DWORD	0x00000000 (0)
SystemPages	REG_DWORD	0x00000000 (0)
WriteWatch	REG_DWORD	0x00000001 (1)

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management

Locate the ClearPageFileAtShutdown entry in the right pane of the window and change its value to `1`. Now restart Windows for the change to take effect, and your swap file will be cleared at shutdown. The only side effect of enabling this is that Windows may take longer to shut down. However, this is very much dependent on your hardware (e.g., disk controller chipset, disk drive speed, processor speed, etc.), since that's what

will govern how long it will take to overwrite your paging file with zeros.

Hack 30 Restrict Applications Available to Users

Prevent your users from running potentially dangerous applications.

Keeping

users from running certain applications isn't so important when you're an

administrator using your own workstation. But when

you're dealing with regular users in an enterprise

network environment, you don't want your users

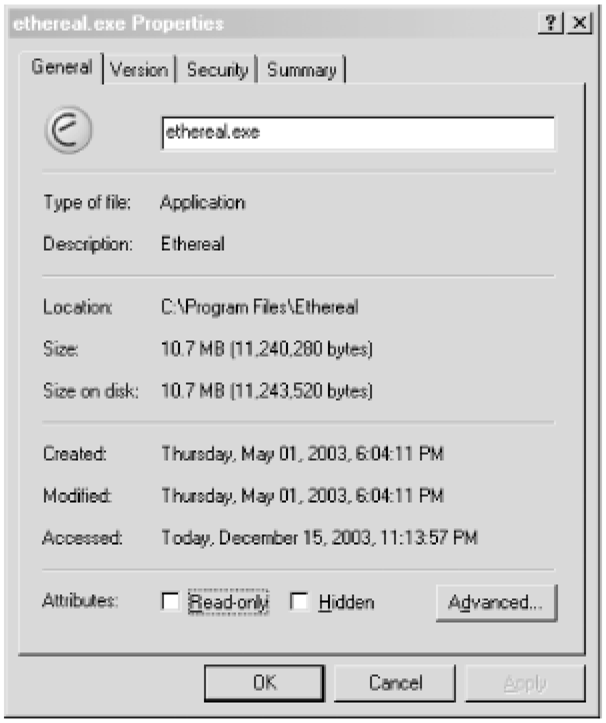
running any nefarious programs. Such programs include those that can break their operating system installation, introduce security holes to their system, or even attack other machines on your network.

There are a couple ways to restrict the applications available to your users. First you can modify the ACLs for a particular program so that users cannot execute it. For example, suppose you have a sniffer installed on a user's machine for

network diagnostic purposes. Access to this program is fine for an administrator, but probably is not appropriate for a normal user. You can prevent normal users from running the program by removing execution permissions for the Users group. To do this, locate the program's executable file and right-click it. Now

click the Properties menu item, and you should see a dialog box like the one shown in Figure 2-9.

Figure 2-9. Properties dialog for ethereal.exe, the Ethernet sniffer


ethereal.exe Properties

General | Version | Security | Summary

ethereal.exe

Type of file: Application
Description: Ethereal

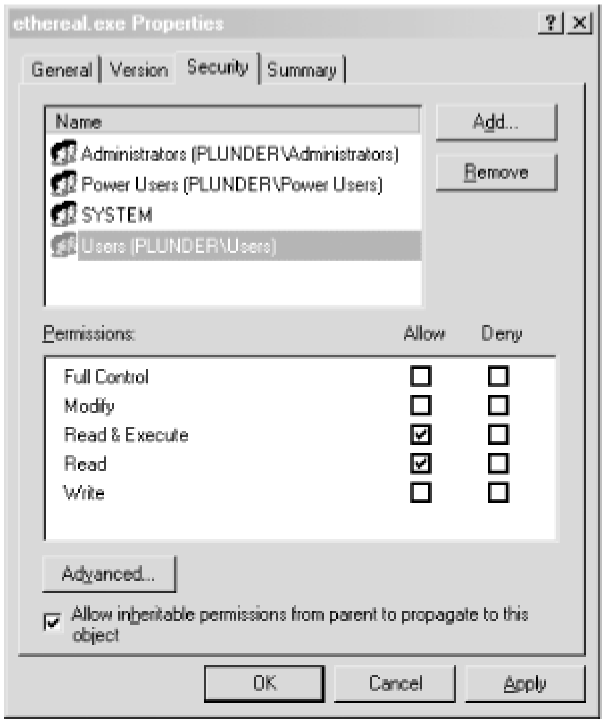
Location: C:\Program Files\Ethereal
Size: 10.7 MB (11,240,280 bytes)
Size on disk: 10.7 MB (11,243,520 bytes)

Created: Thursday, May 01, 2003, 6:04:11 PM
Modified: Thursday, May 01, 2003, 6:04:11 PM
Accessed: Today, December 15, 2003, 11:13:57 PM

Attributes: ☐ Read-only ☐ Hidden

Now click on the Security tab and select the Users group from the list at the top of the dialog. You should now see something similar to Figure 2-10.

Figure 2-10. The Security tab of the ethereal.exe Properties dialog


ethereal.exe Properties

General | Version | Security | Summary

Name
Administrators (PLUNDER\Administrators)
Power Users (PLUNDER\Power Users)
SYSTEM
Users (PLUNDER\Users)

Permissions:	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	☑	☐
Read	☑	☐
Write	☐	☐

☑ Allow inheritable permissions from parent to propagate to this object

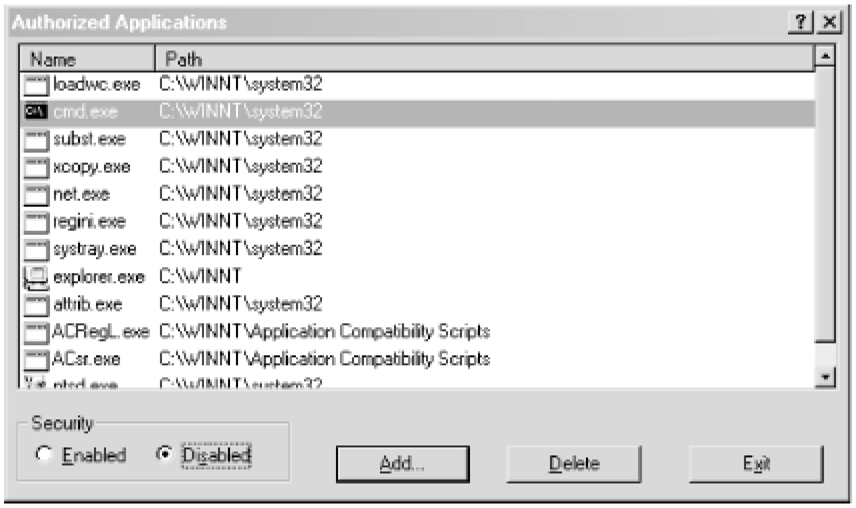
Now click the Deny checkbox that applies to the Read & Execute permission. After clicking the Apply button, anyone that is a member of the Users group will not be able to run the program.

Alternatively, you could also modify the ACL for the directory that the program resides in and disallow read access. This approach could be useful if you want to keep all of your administrative tools under a single folder and restrict access to all of them at once.

If you are running a terminal-server version of Windows, there is another alternative to using ACLs. If you have the Microsoft Windows 2000 resource kit installed, you can use the

AppSec program to disallow program access with just a few clicks. To use *AppSec*, locate its directory and start the program. After the program loads, you will be presented with a list of programs. If the program that you want to disallow from your terminal-service users is on the list, simply click the Disabled radio button. For instance, if you wanted to disable *cmd.exe*, you would see something similar to Figure 2-11.

Figure 2-11. Restricting cmd.exe



Authorized Applications

Name	Path
loadwc.exe	C:\WINNT\system32
cmd.exe	C:\WINNT\system32
subst.exe	C:\WINNT\system32
xcopy.exe	C:\WINNT\system32
net.exe	C:\WINNT\system32
regini.exe	C:\WINNT\system32
systray.exe	C:\WINNT\system32
explorer.exe	C:\WINNT
attrib.exe	C:\WINNT\system32
ACRegL.exe	C:\WINNT\Application Compatibility Scripts
ACsr.exe	C:\WINNT\Application Compatibility Scripts
ntsd.exe	C:\WINNT\system32

Security: Enabled [✓ Disabled]

If the application you want to restrict is not on the list, you can click the Add button and browse for the application. After you have made your choices, click Exit. Before these changes can fully take effect, all users will have to log off of the terminal server.

Chapter 3. Network Security

Hacks #31-53

As we rely more and more on massively interconnected networks, the stability and security of these networks is more vital than ever. The world of business has

adopted information technology to help streamline their processes, increase productivity, and cut costs.

As such, a company's IT infrastructure is

a core asset to many businesses.

Because of this, many businesses would cease to function if disaster (whether natural or digital) were to disrupt their network operations in a significant way.

At the same time, the widespread adoption of the Internet as a global communications medium has also brought computer networks out of the business and academic world and into our personal lives, where it is used not only for entertainment, but also as a means to keep in touch with friends, family, and loved ones.

Although this book as a whole is meant to address network security, the information it contains extends into many other areas. After all, a network is simply a means to connect machines and services together so that they can communicate. This chapter, however,

deals primarily with the security and integrity of the network itself. In this chapter,

you'll learn how to detect and prevent certain types of spoofing attacks that can be used to compromise the core integrity of a TCP/IP Ethernet network at its lowest level. This chapter also includes a great deal of information about firewalls, discussing everything from basic port-based firewalling to MAC-address filtering, and even shows you how to create a gateway that will authenticate machines based on login credentials.

Although it is not always a direct security threat, network reconnaissance is often a precursor to an attack. In this chapter, you'll learn how to fool those who are trying to gather information about the hosts on your network, as well as ways to detect eavesdroppers who are monitoring your network for juicy bits of information.

```
01:20:14.833350 arp who-has 192.168.0.66 tell 192.168.0.62

01:20:14.833421 arp reply 192.168.0.66 is-at 0:0:d1:1f:3f:f1
```

```
# <b>./configure && make && make install</b>
```

```
<span class="docEmphBold">arpwatch -i iface</span>
```

```
Nov 1 00:39:08 zul arpwatch: new station 192.168.0.65 0:50:ba:85:85:ca

Nov 1 01:03:23 zul arpwatch: changed ethernet address 192.168.0.65
0:e0:81:3:d8:8e

(0:50:ba:85:85:ca)



Nov 1 01:03:23 zul arpwatch: flip flop 192.168.0.65 0:50:ba:85:85:ca
(0:e0:81:3:d8:8e)

Nov 1 01:03:25 zul arpwatch: flip flop 192.168.0.65 0:e0:81:3:d8:8e
(0:50:ba:85:85:ca)
```

In this case, the initial entry is from the first fraudulent ARP response that was received, and the subsequent two are from a race condition between the fraudulent and authentic responses.

To make it easier to deal with multiple Arpwatch installs on a switched environment, you can send the log messages to a central syslogd [Hack #54], aggregating all the output into one place. However, due to the fact that your machines can be manipulated by the same attacks that Arpwatch is looking for, it would be wise to use static ARP table entries [Hack #32] on your syslog server, as well as all the hosts running Arpwatch.

```
<b>arp -s</b> <span class="docEmphBoldItalic">ipaddr macaddr</span>
```

```
# <b>arp -s 192.168.0.65 00:50:ba:85:85:ca</b>
```

```
#!/usr/bin/perl
```

```
#

# gen_ethers.pl <from ip> <to ip>
```

```
#

my ($start_1, $start_2, $start_3, $start_4) = split(/\./, $ARGV[0], 4); my
($end_1, $end_2, $end_3, $end_4) = split(/\./, $ARGV[1], 4); my
$ARP_CMD="/sbin/arp -n";

for(my $oct_1 = $start_1; $oct_1 <= $end_1 && $oct_1 <= 255; $oct_1++
){

   for(my $oct_2 = $start_2; $oct_2 <= $end_2 && $oct_2 <= 255;
$oct_2++){

   for(my $oct_3 = $start_3; $oct_3 <= $end_3 && $oct_3 <= 255;
$oct_3++){

   for(my $oct_4 = $start_4; $oct_4 <= $end_4 && $oct_4 < 255;
$oct_4++){

   system("ping -c 1 -W 1 $oct_1.$oct_2.$oct_3.$oct_4 > /dev/null 2>&1");
my $ether_addr = `$ARP_CMD $oct_1.$oct_2.$oct_3.$oct_4 | egrep
'HWaddress|

(incomplete)' | awk '{print \$3}'`;

   chomp($ether_addr);

   if(length($ether_addr) == 17){

   print("$ether_addr\t$oct_1.$oct_2.$oct_3.$oct_4\n"); }

   }

   }

   }

}
```

```
# <b>./gen_ethers 192.168.1.1 192.168.1.255 > /etc/ethers</b>

# <b>arp -f /root/arp_entries</b>
```

This script isn't perfect, but it can save a lot of time when creating static ARP table entries for the hosts on your network. Once you've generated the file with the MAC/IP pairings, you can copy it to the other hosts and add an `arp` command to the system startup scripts, to automatically load them at boot time. The main downside to using this method is that all the devices on your network need to be powered on when the script runs; otherwise, they will be missing from the list. In addition, if the machines on your network change frequently, you'll have to regenerate and distribute the file often, which may be more trouble than it's worth. But for servers and devices that never change their IP or MAC address, this method can protect your machines from ARP poisoning attacks.

```
# <b>iptables -P INPUT DROP</b>


# <b>iptables -P FORWARD DROP</b>

# <b>iptables -P INPUT -i lo -j ACCEPT</b>

# <b>iptables -P OUTPUT -o lo -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b> -d 192.168.1.20 --dport 80 -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b> -d 192.168.1.21 --dport 25 -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b> -d 192.168.1.21 --dport 110 -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b> -d 192.168.1.21 --dport 143 -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b> -d 192.168.1.21 --dport 993 -j ACCEPT</b>

# <b>iptables -A FORWARD -m state --state NEW -p tcp \</b>

<b>-d 192.168.1.21 --dport 53 -j ACCEPT</b>

# <b>iptables -A FORWARD -p udp -d 192.168.1.18 --dport 53 -j
ACCEPT</b>

# <b>iptables -A FORWARD -p udp -s 192.168.1.18 --sport 53 -j
ACCEPT</b>
```

```
# <b>iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT</b>
```

```
# <b>iptables -A FORWARD -m state --state NEW -i eth1 -j ACCEPT</b>
```

```
# <b>echo 1 > /proc/sys/net/ipv4/ip_forward</b>
```

```
net.ipv4.ip_forward=1
```

```
# <b>echo 1 > /proc/sys/net/ipv4/conf/default/rp_filter</b>
```

```
net.ipv4.conf.all.rp_filter=1
```

```
# <b>/sbin/service iptables save</b>
```

```
<b># /etc/init.d/iptables save_active</b>
```

When the machine reboots, your iptables configuration will be automatically restored.

Hack 34 Firewall with OpenBSD's PacketFilter

Use OpenBSD's firewalling features to protect your network.

PacketFilter, commonly known as PF, is the firewalling system available in OpenBSD. While it is a relatively new addition to the operating system, it has already surpassed IPFilter, the system it has replaced, in both features and flexibility. PF shares many features with Linux's Netfilter. Although Linux's Netfilter is more easily extensible with modules, PF outshines it in its traffic normalization capabilities and enhanced logging features.

To communicate with the kernel portion of PF, we need to use the `pfctl` command. Unlike the `iptables` command that is used with Linux's Netfilter, it is not used to specify individual rules, but instead uses its own configuration and rule specification language. To actually configure PF, we must edit */etc/pf.conf*. PF's rule specification language is actually very powerful, flexible, and easy to use. The *pf.conf* file is split up into seven sections, each of which contains a particular type of rule. Not all sections need to be usedif you don't need a specific type of rule, that section can simply be left out of the file.

The first section is for macros. In this section you can specify variables to hold either single values or lists of values for use in later sections of the configuration file. Like an environment variable or a programming-language identifier, macros must start with a letter and also may contain digits and underscores.

Here are some example macros:

```
EXT_IF="de0"

INT_IF="de1"

RFC1918="{ 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }"
```

A macro can be referenced later by prefixing it with the `$` character:

```
block drop quick on $EXT_IF from any to $RFC1918
```

The second section allows you to specify tables of IP addresses to use in later rules. Using tables for lists of IP addresses is much faster than using a macro, especially for large numbers of IP addresses, because when a macro is used in a rule, it will expand to multiple rules, with each one matching on a single value contained in the macro. Using a table adds just a single rule when it is expanded.

Rather than using the macro from our previous example, we can define a table to hold the nonroutable RFC 1918 IP addresses:

```
table <rfc1918> const { 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
```

The `const` keyword ensures that this table cannot be modified once it has been created. Tables are specified in a rule in the same way that they were created:

```
block drop quick on $EXT_IF from any to <rfc1918>
```

You can also load a list of IP addresses into a table by using the `file` keyword:

```
table <spammers> file "/etc/spammers.table"
```

If you elect not to use the `const` keyword, then you can add addresses to a table by running a command such as this:

```
pfctl -t spammers -T add 10.1.1.1
```

Additionally, you can delete an address by running a command like this:

```
pfctl -t spammers -T delete 10.1.1.1
```

To list the contents of a table, you can run:

```
pfctl -t spammers -T show
```

In addition to IP addresses, hostnames may also be specified. In this case, all valid addresses returned by the resolver will be inserted into the table.

The next section of the configuration file contains options that affect the behavior of PF. By modifying options, we can control session timeouts, defragmentation timeouts, state-table transitions, statistic collection, and other behaviors. Options are specified by using the `set` keyword. The number of options is too numerous to discuss all of them in any meaningful detail; however, we will discuss the most pertinent and useful ones.

One of the most important options is `block-policy`. This option specifies the default behavior of the `block` keyword and can be configured to silently drop matching packets by specifying `drop`. Alternatively, `return` may be used, to specify that packets matching a block rule will generate a TCP reset or an ICMP unreachable packet, depending on whether the triggering packet is TCP or UDP. This is similar to the REJECT target in Linux's Netfilter.

For example, to have PF drop packets silently by default, add a line like this to */etc/pf.conf*:

```
set block-policy drop
```

In addition to setting the `block-policy`, additional statistics such as packet and byte counts can be collected for an interface. To enable this for an interface, add a line similar to this to the configuration file:

```
set loginterface de0
```

However, these statistics can only be collected on a single interface at a time. If you do not want to collect any statistics, you can replace the interface name with the `none` keyword.

To better utilize resources on busy networks, we can also modify the session-timeout values. Setting this to a low value can help improve the performance of the firewall on high-traffic networks, but at the expense of dropping valid idle connections.

To set the session timeout (in seconds), put a line similar to this in */etc/pf.conf*:

```
set timeout interval 20
```

With this setting in place, any TCP connection that is idle for 20 seconds will automatically be reset.

PF can also optimize performance on low-end hardware by tuning its memory use regarding how many states may be stored at any one time or how many fragments may reside in memory for fragment reassembly. For example, to set the number of states to 20,000 and the number of entries used by the fragment reassembler to 15,000, we could put this in our *pf.conf*:

```
set limit states 20000

set limit frags 15000
```

Alternatively, we could combine these entries into a single one, like this:

```
set limit { states 20000, frags 15000 }
```

Moving on, the next section is for traffic normalization rules. Rules of this type ensure that packets passing through the firewall meet certain criteria regarding fragmentation, IP IDs, minimum TTLs, and other attributes of a TCP datagram. Rules in this section are all prefixed by the `scrub` keyword. In general, just putting `scrub all` is fine. However, if necessary, we can get quite detailed in specifying what we want normalized and how we want to normalize it. Since we can use PF's general filtering-rule syntax to determine what types of packets a scrub rule will match, we can normalize packets with a great deal of control.

One of the more interesting possibilities is to randomize all IP IDs in the packets leaving your network for the outside world. In doing this, we can make sure that passive operating system determination methods based on IP IDs will break when trying to figure out the operating system of a system protected by the firewall. Because such methods depend on analyzing how the host operating system increments the IP IDs in its outgoing packets, and our firewall ensures that the IP IDs in all the packets leaving our network are totally random, it's pretty hard to match them against a known pattern for an operating system. This also helps to prevent enumeration of machines in a network address translated (NAT) environment. Without random IP IDs, someone outside the network can perform a statistical analysis of the IP IDs being emitted by the NAT gateway in order to count the number of machines on the private network. Randomizing the IP IDs defeats this kind of attack.

To enable random ID generation on an interface, put a line such as this in */etc/pf.conf*:

```
scrub out on de0 all random-id
```

We can also use the `scrub` directive to reassemble fragmented packets before forwarding them to their destinations. This helps prevent specially fragmented packets (such as packets that overlap) from evading intrusion-detection systems that are sitting behind the firewall.

To enable fragment reassembly on all interfaces, simply put the following line in the configuration file:

```
scrub fragment reassemble
```

If we want to limit reassembly to just a single interface, we can change this to:

```
scrub in on de0 all fragment reassemble
```

This will enable fragment reassembly for the `de0` interface.

The next two sections of the *pf.conf* file involve packet queuing and address translation, but since this hack focuses on packet filtering, we'll skip these. This brings us to the last section, which contains the actual packet-filtering rules. In general, the syntax for a filter rule can be defined by the following:

```
action direction [log] [quick] on int [af] [proto protocol] \

  from src_addr [port src_port] to dst_addr [port dst_port] \

  [tcp_flags] [state]
```

In PF, a rule can have only two actions: block and pass. As discussed previously, the block policy affects the behavior of the block action. However, this can be modified for specific rules by specifying it along with an action, such as `block drop` or `block return`. Additionally, `block return-icmp` can be used, which will return an ICMP unreachable message by default. An ICMP type can be specified as well, in which case that type of ICMP message will be returned.

For most purposes, we want to start out with a default deny policy; that way we can later add rules to allow the specific traffic that we want through the firewall.

To set up a default deny policy for all interfaces, put the following line in */etc/pf.conf*:

```
block all
```

Now we can add rules to allow traffic through our firewall. First we'll keep the loopback interface unfiltered. To accomplish this, we'll use this rule:

```
pass quick on lo0 all
```

Notice the use of the `quick` keyword. Normally PF will continue through our rule list even if a rule has already allowed a packet to pass, in order to see whether a more specific rule that appears later on in the configuration file will drop the packet. The use of the `quick` keyword modifies this behavior and causes PF to stop processing the packet at this rule if it matches the packet and to take the specified action. With careful use, this can greatly improve the performance of a ruleset.

To prevent external hosts from spoofing internal addresses, we can use the `antispoof` keyword:

```
antispoof quick for $INT_IF inet
```

Next we'll want to block any packets from entering or leaving our external interface that have a nonroutable RFC 1918 IP address. Such packets, unless explicitly allowed later, would be caught by our default deny policy. However, if we use a rule to specifically match these packets and use the `quick` keyword, we can increase performance by adding a rule like this:

```
block drop quick on $EXT_IF from any to <rfc1918>
```

If we wanted to allow traffic into our network destined for a web server at 192.168.1.20, we could use a rule like this:

```
pass in on $EXT_IF proto tcp from any to 192.168.1.20 port 80 \

  modulate state flags S/SA
```

This will allow packets destined to TCP port 80 at 192.168.1.20 only if they are establishing a new connection (i.e., the `SYN` flag is set), and will enter the connection into the state table. The `modulate` keyword ensures that a high-quality initial sequence number is generated for the session, which is important if the operating system in use at either end of the connection uses a poor algorithm for generating its ISNs.

Similarly, if we wanted to pass traffic to and from an email server at the IP address 192.168.1.21, we could use this rule:

```
pass in on $EXT_IF proto tcp from any to 192.168.1.21 \

  port { smtp, pop3, imap2, imaps } modulate state flags S/SA
```

Notice that multiple ports can be specified for a rule by separating them with commas and enclosing them in curly braces. We can also use service names, as defined in */etc/services*, instead of specifying the service's port number.

To allow traffic to a DNS server at 192.168.1.18, we can add a rule like this:

```
pass in on $EXT_IF proto tcp from any to 192.168.1.18 port 53 \

  modulate state flags S/SA
```

This still leaves the firewall blocking UDP DNS traffic. To allow this through, add this rule:

```
pass in on $EXT_IF proto udp from any to 192.168.1.18 port 53 \

  keep state
```

Notice here that even though this is a rule for UDP packets we have still used the `state` keyword. In this case, PF will keep track of the connection using the source and destination IP address and port pairs. Also, since UDP datagrams do not contain sequence numbers, the `modulate` keyword is not applicable. We use `keep state` instead, which is how to specify stateful inspection when not modulating ISNs. In addition, since UDP datagrams do not contain flags, we simply omit them.

Now we'll want to allow connections initiated from the internal network to pass through the firewall. To do this, we'll need to add the following rules to let the traffic into the internal interface of the firewall:

```
pass in on $INT_IF from $INT_IF:network to any

pass out on $INT_IF from any to $INT_IF:network

pass out on $EXT_IF proto tcp all modulate state flags S/SA

pass out on $EXT_IF proto { icmp, udp } all keep state
```

As you can see, OpenBSD has a very powerful and flexible firewalling system. There are too many features and possibilities to discuss here. For more information, you can look at the excellent PF documentation available online or the *pf.conf* manpage.

```
pass in quick on wi0 proto { tcp, udp } from $user_ip to any \
   keep state flags S/SA
pass in quick on wi0 proto tcp from $user_ip to any \
   port { smtp, www, https, pop3, pop3s, imap, imaps } \
   keep state flags S/SA
pass in quick on wi0 proto udp from $user_ip to any port domain
nat-anchor authpf


rdr-anchor authpf


binat-anchor authpf
```

```
anchor authpf
```

```
Hello andrew, You are authenticated from host "192.168.0.61"
```

```
Dec 3 22:36:31 zul authpf[15058]: allowing 192.168.0.61, \
   user andrew
Dec 3 22:47:21 zul authpf[15058]: removed 192.168.0.61, \
   user andrew- duration 650 seconds
```

```
ClientAliveInterval 15


ClientAliveCountMax 3
```

This will ensure that the SSH daemon will send a request for a client response 15 seconds after it has received no data from the client. The `ClientAliveCountMax` option specifies that this can happen three times without a response before the client is disconnected. Thus, after a client has become unresponsive, it will be disconnected after 45 seconds. These keepalive packets are sent automatically by the SSH client software and don't require any intervention on the part of the user.

Authpf is very powerful in its flexibility and integration with PF, OpenBSD's native firewalling system. It is easy to set up and has very little performance overhead, since it relies on SSH and the operating system to do authentication and manage sessions.

Hack 36 Firewall with Windows

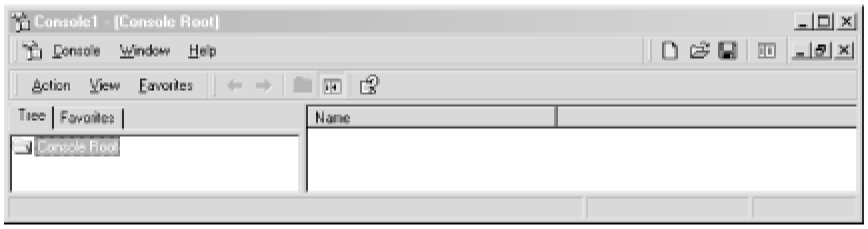
Yes, you can use Windows as a firewall.

You may not know it, but Windows has a very capable firewall built right in.

To access it, run the Microsoft Management Console. You can do this by opening up a Run dialog, typing `mmc`, and

clicking the OK button. After the program loads, you should see something similar to Figure 3-1.

Figure 3-1. The Microsoft Management Console



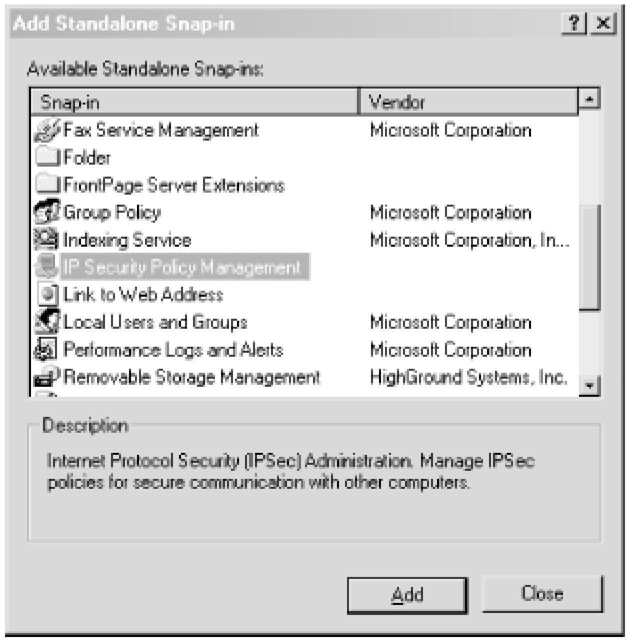
Click on the Console menu and select the "Add/Remove Snap-in..." menu item. Next you should be presented

with a dialog that has an Add button at the bottom. After clicking the Add button, you should see a dialog box with a list of available snap-ins. Scroll through the list and locate the item titled

IP Security Policy Management. After you've selected this, the dialog box should look

like Figure 3-2.

Figure 3-2. Adding the IP Security Policy Management snap-in



Available Standalone Snap-ins:

Snap-in	Vendor
Fax Service Management	Microsoft Corporation
Folder	
FrontPage Server Extensions	
Group Policy	Microsoft Corporation
Indexing Service	Microsoft Corporation, In...
IP Security Policy Management	
Link to Web Address	
Local Users and Groups	Microsoft Corporation
Performance Logs and Alerts	Microsoft Corporation
Removable Storage Management	HighGround Systems, Inc.

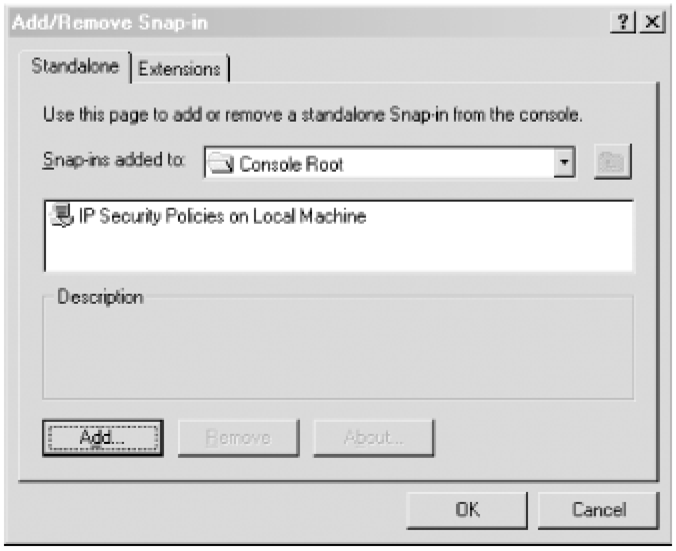
Description

Internet Protocol Security (IPSec) Administration. Manage IPSec policies for secure communication with other computers.

Now click the Add button. You'll be presented with a dialog asking whether you want the snap-in to manage the local computer or a domain. Determine whether you want to apply the filtering settings to just the local computer or the entire domain, and click the Finish button. Click the Close button in the Add Standalone Snap-in list dialog as shown in Figure 3-2. You should now see the IP Security Policies snap-in listed in the Add/Remove Snap-in dialog, as shown in Figure 3-3. Click the OK button and you'll be returned to the original Management

Console window. You should now see the IP Security Policies snap-in listed in the window.

Figure 3-3. The Add/Remove Snap-in dialog with the IP Security Policies snap-in loaded



Use this page to add or remove a standalone Snap-in from the console.

Snap-ins added to: Console Root

IP Security Policies on Local Machine

Description

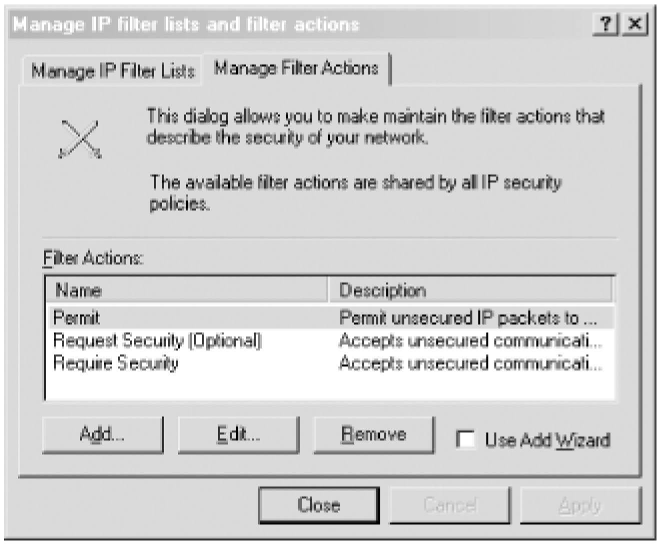
Before setting up firewall rules, you'll need to create a block action for them to use. To do this, right-click the IP

Security Policies icon and select the "Manage IP

filter lists and filter actions" item. After the

dialog appears, click on the Manage Filter Actions tab. You should now see something similar to Figure 3-4.

Figure 3-4. The Manage Filter Actions tab



This dialog allows you to make maintain the filter actions that describe the security of your network.

The available filter actions are shared by all IP security policies.

Filter Actions:

Name	Description
Permit	Permit unsecured IP packets to ...
Request Security (Optional)	Accepts unsecured communicati...
Require Security	Accepts unsecured communicati...

If the Use Add Wizard checkbox is not checked, be sure to check it.

Now click the Add button. Click the Next button after the wizard dialog opens. Then type "Block" for

name of the new filter action. For the description, type

"Blocks Access" or something

similarly appropriate. After filling those in, click the Next button.

Now click the Block radio button, and then click the Next button once again. After that, click the Finish button. You should now see the new filter action in the list that was shown in Figure 3-3. You may now click the Close button.

Now you can set up the firewall rules. Right-click the

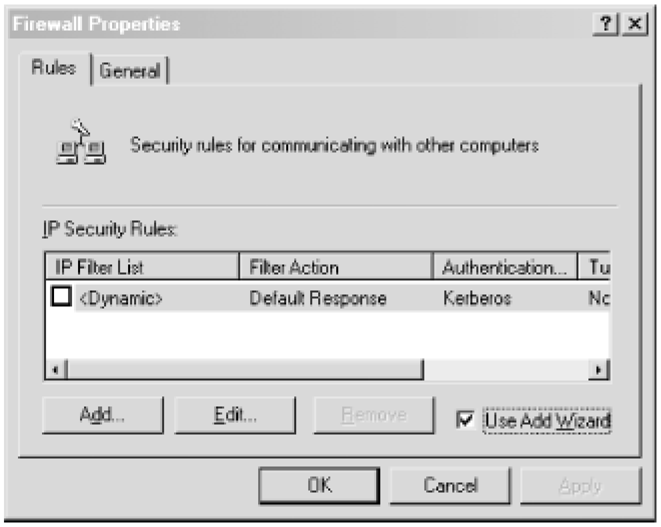
security policy icon and select the Create IP Security Policy item. This will bring up a wizard. Click the Next button and fill in the Name and Description; a good choice for both of them would be "Firewall". After

filling those in, click the Next button. You should now see a checkbox labeled "Activate the default response

rule". Uncheck this box and then click the Next

button. After that, click the Finish button. You should now see a dialog called Firewall Properties, as shown in Figure 3-5.

Figure 3-5. The Firewall Properties dialog

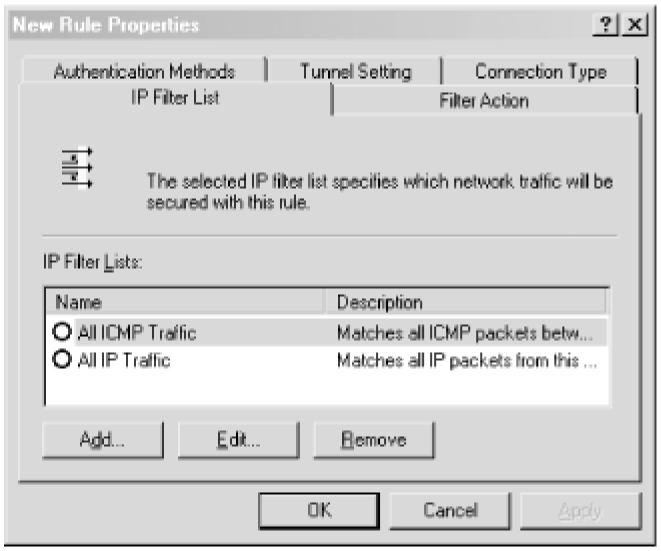


IP Filter List	Filter Action	Authentication...	Tu
☐ <Dynamic>	Default Response	Kerberos	Nc

☑ Use Add Wizard

To create a new filtering rule, uncheck the Use Add Wizard box and click the Add button. You should now see a dialog box that looks like Figure 3-6.

Figure 3-6. Adding a new rule



Name	Description
○ All ICMP Traffic	Matches all ICMP packets betw...
○ All IP Traffic	Matches all IP packets from this ...

To select the IP addresses to match on, click the Add button in the IP Filter List tab. This will also let you define ports and protocols to match on. After you have selected the IP addresses and ports you want the rule to apply to, click the Filter Action tab and choose your selections from the list of actions.

```
rdr on $<tt><i>INT_IF</i></tt> inet proto { tcp, udp } from $<tt>
<i>INT_IF</i></tt>:network to any port 53



-> $<tt><i>DNS_SERVER</i></tt> port 53

rdr on $<tt><i>INT_IF</i></tt> inet proto tcp from $<tt><i>INT_IF</i>
</tt>:network to any port 25 -> $<tt><i>SMTP_HOST</i></tt> port 25
```

Egress filtering can also prevent IP spoofing. By filtering on your external interface at the border of your network, you can verify that packets leaving your network have source addresses that match your address space. By filtering all other traffic, you can ensure that any IP spoofing attack performed from your network or routed through it will be dropped before the packets are able to leave.

```
# <b>perl -MCPAN -e "install Net::RawIP"</b>
```

```
# <b>perl -MCPAN -e "install Net::PcapUtils"</b>
```

```
# <b>perl -MCPAN -e "install NetPacket"</b>
```

<tt><i>source addr</i></tt>:<tt><i>source port</i></tt>:<tt><i>dest addr</i></tt>:<tt><i>dest port</i></tt>:<tt><i>flags</i></tt>:<tt><i>proto</i></tt>:<tt><i>tos</i></tt>

<tt><i>source addr</i></tt>::<tt><i>dest addr</i></tt>:::ICMP:<tt><i>type</i></tt>:<tt><i>code</i></tt>

```
192.168.1.10:1025:10.1.1.1:1-1025:S:TCP:0


stop_signal=192.168.1.10:1025:10.1.1.1:22:S:TCP:0
```

```
# <b>./ftestd -i eth0</b>
```

```
# <b>./ftest -f ftest.conf</b>
```

```
# <b>./freport ftest.log ftestd.log</b>
```

```
Authorized packets:

-------------------


22 - 192.168.1.10:1025 > 10.1.1.1:22 S TCP 0


25 - 192.168.1.10:1025 > 10.1.1.1:25 S TCP 0
```

```
80 - 192.168.1.10:1025 > 10.1.1.1:80 S TCP 0

Modified packets (probably NAT):

--------------------------------

Filtered or dropped packets:

----------------------------

1 - 192.168.1.10:1025 > 10.1.1.1:1 S TCP 0

2 - 192.168.1.10:1025 > 10.1.1.1:2 S TCP 0

3 - 192.168.1.10:1025 > 10.1.1.1:3 S TCP 0
```

If you are using a stateful firewall and want to test this functionality, you can also specify packets that have flags other than SYN set. For instance, if the previous example had used ACK or some other flag instead of SYN, it would be dropped by the firewall since only packets with the SYN flag set are used to initiate connections.

It's a good idea to run *ftest* each time you make changes to your firewall, or periodically just to make sure that your firewall works as you expect it to. While complex rulesets on your firewall can sometimes make it difficult to predict exactly how it will behave, *ftest* will tell you with good authority exactly what kind of traffic is permitted.

```
iptables -A FORWARD -m state --state NEW \

   -m mac --mac-source 00:DE:AD:BE:EF:00 -j ACCEPT
```

```
$ <b>ifconfig eth0</b>

eth0 Link encap:Ethernet HWaddr 00:0C:29:E2:2B:C1

   inet addr:192.168.0.41 Bcast:192.168.0.255 Mask:255.255.255.0

   UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

   RX packets:132893 errors:0 dropped:0 overruns:0 frame:0

   TX packets:17007 errors:0 dropped:0 overruns:0 carrier:0

   collisions:0 txqueuelen:100

   RX bytes:46050011 (43.9 Mb) TX bytes:1601488 (1.5 Mb) Interrupt:10
Base address:0x10e0
```

```
$ <b>ping -c 1 192.168.0.61</b>

$ <b>/sbin/arp 192.168.0.61 | awk '{print $3}'</b>
```

```
#!/bin/sh

ping -c $1 >/dev/null && /sbin/arp $1 | awk '{print $3}' \ | grep -v
Hwaddress
```

When implementing MAC address filtering, please be aware that it is not foolproof. Under many circumstances, it is quite trivial to change the MAC address that an interface uses by simply instructing the driver to do so. It is also possible to send out link-layer frames with forged MAC addresses by using raw link-layer sockets. Thus, MAC address filtering should only be considered as an additional measure that you can use to protect your network. Treat MAC filtering more as a "Keep Out" sign, rather than a good deadbolt.

```
# <b>nmap -O puffy</b> Starting nmap 3.48 (
http://www.insecure.org/nmap/ ) at 2003-12-02 19:14 MST

Interesting ports on puffy (192.168.0.42): (The 1653 ports scanned but not
shown below are in state: closed)
```

PORT STATE SERVICE

```
13/tcp open daytime

22/tcp open ssh

37/tcp open time

113/tcp open auth

Device type: general purpose

Running: OpenBSD 3.X

OS details: OpenBSD 3.0 or 3.3

Nmap run completed -- 1 IP address (1 host up) scanned in 24.873 seconds

set block-policy return

block in log quick proto tcp flags FUP/WEUAPRSF

block in log quick proto tcp flags WEUAPRSF/WEUAPRSF

block in log quick proto tcp flags SRAFU/WEUAPRSF

block in log quick proto tcp flags /WEUAPRSF

block in log quick proto tcp flags SR/SR

block in log quick proto tcp flags SF/SF

# <b>ifconfig pflog0 up</b> # <b>tcpdump -n -i pflog0</b>

# <b>nmap -O puffy</b> Starting nmap 3.48 (
http://www.insecure.org/nmap/ ) at 2003-12-02 22:56 MST
```

Interesting ports on puffy (192.168.0.42): (The 1653 ports scanned but not shown below are in state: closed)

```
PORT STATE SERVICE

13/tcp open daytime

22/tcp open ssh

37/tcp open time

113/tcp open auth

No exact OS matches for host (If you know what OS is running on it, see
http://www.insecure.org/cgi-bin/nmap-submit.cgi).

TCP/IP fingerprint:

SInfo(V=3.48%P=i686-pc-linux-
gnu%D=12/2%Time=3FCD7B3F%O=13%C=1)
TSeq(Class=TR%IPID=RD%TS=2HZ)

T1(Resp=Y%DF=Y%W=403D%ACK=S++%Flags=AS%Ops=MNWNNT
) T2(Resp=Y%DF=Y%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
T4(Resp=Y%DF=Y%W=4000%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=134%RID=E%RIPCK
=F%UCK=E%ULEN=134%DAT=E) Nmap run completed -- 1 IP address
(1 host up) scanned in 27.028 seconds
```

As you can see, this time the attempt was unsuccessful. But if you are feeling particularly devious, simply confusing Nmap attempts may not be enough. What if you want to actually trick would-be attackers into believing that a server is running a different OS entirely? For example, this could be useful when setting up a honeypot [Hack #94] to attract miscreants away from your critical servers. If this sounds like fun to you, read on.

```
# <b>cd /usr/src/linux</b> # <b>patch -p1 < \ </b> <b>../ippersonality-
20020819-2.4.19/patches/ippersonality-20020819-linux-2.4.19.diff</b>

# <b>find ./ -name \*.rej</b>

# <b>make dep && make clean</b> # <b>make bzImage && make
modules</b> # <b>cp arch/i386/boot/bzImage /boot/vmlinuz</b> #
<b>make modules_install</b>

# <b>tar xfj iptables-1.2.2.tar.bz2</b> # <b>cd iptables-1.2.2</b> #
<b>patch -p1 < \</b> <b>../ippersonality-20020819-
2.4.19/patches/ippersonality-20020427-iptables-\1.2.2.diff </b> patching
file pers/Makefile patching file pers/example.conf patching file
pers/libipt_PERS.c patching file pers/pers.h

patching file pers/pers.l

patching file pers/pers.y

patching file pers/pers_asm.c patching file pers/perscc.c # <b>make
KERNEL_DIR=/usr/src/linux && make install</b>

LIBDIR:=/lib

BINDIR:=/sbin

MANDIR:=/usr/share/man

INCDIR:=/usr/include

# <b>nmap -O colossus</b> Starting nmap 3.48 (
http://www.insecure.org/nmap/ ) at 2003-12-12 18:36 MST

Interesting ports on colossus (192.168.0.64): (The 1651 ports scanned but
not shown below are in state: closed)
```

PORT STATE SERVICE

22/tcp open ssh

25/tcp open smtp

111/tcp open rpcbind

139/tcp open netbios-ssn

505/tcp open mailbox-lm

631/tcp open ipp

Device type: general purpose Running: Linux 2.4.X|2.5.X

OS details: Linux Kernel 2.4.0 - 2.5.20

Uptime 3.095 days (since Tue Dec 9 16:19:55 2003) Nmap run completed -- 1 IP address (1 host up) scanned in 7.375 seconds

# <b>iptables -t mangle -A PREROUTING -d 192.168.0.64 -j PERS \</b> <b>--tweak dst --local --conf /etc/personalities/macos9.conf</b> # <b>iptables -t mangle -A OUTPUT -s 192.168.0.64 -j PERS \</b> <b>--tweak src --local --conf /etc/personalities/macos9.conf</b>

# <b>nmap -O colossus</b> Starting nmap 3.48 (http://www.insecure.org/nmap/) at 2003-12-12 18:47 MST

Interesting ports on colossus (192.168.0.64): (The 1651 ports scanned but not shown below are in state: closed)

```
PORT STATE SERVICE

22/tcp open ssh

25/tcp open smtp

111/tcp open rpcbind

139/tcp open netbios-ssn

505/tcp open mailbox-lm

631/tcp open ipp

Device type: general purpose Running: Apple Mac OS 9.X

OS details: Apple Mac OS 9 - 9.1

Uptime 3.095 days (since Tue Dec 9 16:19:55 2003) Nmap run completed -
- 1 IP address (1 host up) scanned in 5.274 seconds
```

You can of course emulate other operating systems that aren't provided with the IP Personality package. All you need is a copy of Nmap's operating system fingerprints file, *nmap-os-fingerprints*, and then you can construct your own IP Personality configuration file for any operating system Nmap knows about.

```
$ <b>nmap rigel</b> Starting nmap 3.48 ( http://www.insecure.org/nmap/ )
at 2003-12-15 17:42 MST

Interesting ports on rigel (192.168.0.61): (The 1595 ports scanned but not
shown below are in state: filtered)
```

PORT STATE SERVICE

```
7/tcp open echo
9/tcp open discard
13/tcp open daytime
19/tcp open chargen
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
25/tcp open smtp
37/tcp open time
79/tcp open finger
111/tcp open rpcbind
512/tcp open exec
513/tcp open login
514/tcp open shell
587/tcp open submission
4045/tcp open lockd
7100/tcp open font-service
32771/tcp open sometimes-rpc5
```

```
32772/tcp open sometimes-rpc7

32773/tcp open sometimes-rpc9

32774/tcp open sometimes-rpc11

32775/tcp open sometimes-rpc13

32776/tcp open sometimes-rpc15

32777/tcp open sometimes-rpc17

Nmap run completed -- 1 IP address (1 host up) scanned in 75.992 seconds
```

```
nmap 192.168.0.1-254

nmap 192.168.0.0/24
```

```
# <b>nmap -sS -O rigel</b> Starting nmap V. 3.00 (
www.insecure.org/nmap/ ) Interesting ports on rigel.nnc (192.168.0.61):
(The 1578 ports scanned but not shown below are in state: filtered) Port
State Service

7/tcp open echo

9/tcp open discard

13/tcp open daytime

19/tcp open chargen

21/tcp open ftp

22/tcp open ssh

23/tcp open telnet

25/tcp open smtp
```

```
37/tcp open time

79/tcp open finger

111/tcp open sunrpc

512/tcp open exec

513/tcp open login

514/tcp open shell

587/tcp open submission

7100/tcp open font-service

32771/tcp open sometimes-rpc5

32772/tcp open sometimes-rpc7

32773/tcp open sometimes-rpc9

32774/tcp open sometimes-rpc11

32775/tcp open sometimes-rpc13

32776/tcp open sometimes-rpc15

32777/tcp open sometimes-rpc17

Remote operating system guess: Solaris 9 Beta through Release on SPARC

Uptime 44.051 days (since Sat Nov 1 16:41:50 2003) Nmap run completed
-- 1 IP address (1 host up) scanned in 166 seconds

# <b>nmap -sS -O -oX scandata.xml rigel</b>
```

<port protocol="tcp" portid="22"> <state state="open" /> <service
name="ssh" method="table" conf="3" /> </port>

Nmap is a powerful tool. By using its XML output capabilities, a little bit of scripting, and a database, you can create an even more powerful tool that can monitor your network for unauthorized services and machines.

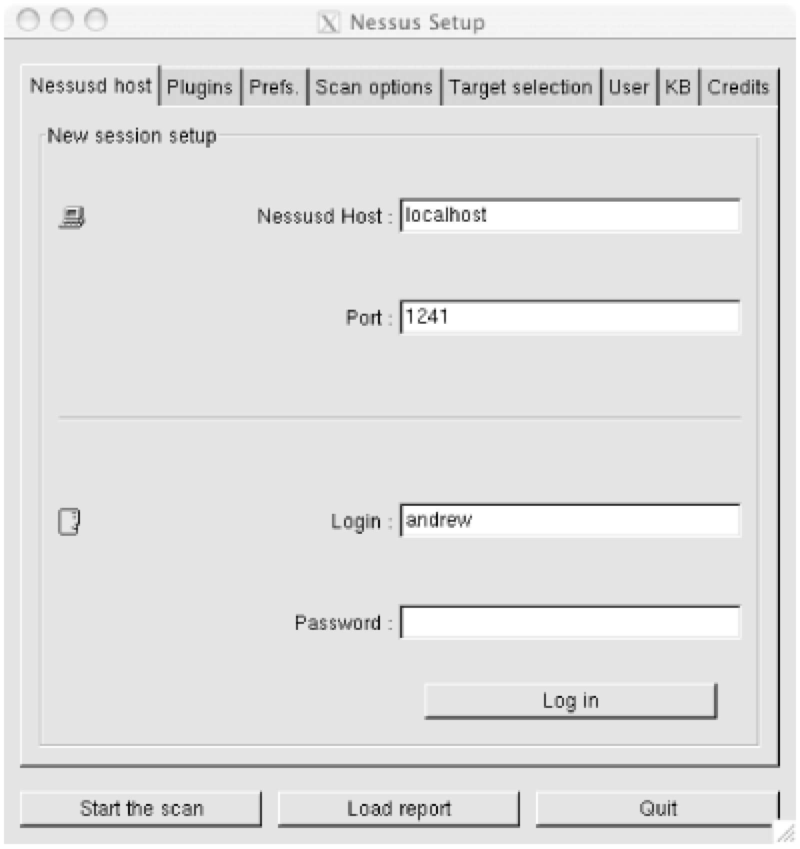
```
$ <b>lynx -source http://install.nessus.org | sh</b>
```

```
cert_file=/etc/ssl/nessus.key


key_file=/etc/ssl/nessus.crt


ca_file=/etc/ssl/ca.crt

pem_password=mypassword
```

```
# <b>/usr/local/sbin/nessusd -D</b>
```

Now you can start the *Nessus* client and connect to the server. There are several *Nessus* clients available, including a command-line interface, an X11 application, and a Windows client. The figures in this hack show the X11 interface. You can start the client by simply typing `nessus`. After you've done that, you should see a window like the one shown in Figure 3-8.

Figure 3-8. Nessus client setup



Nessusd Host : localhost

Port : 1241

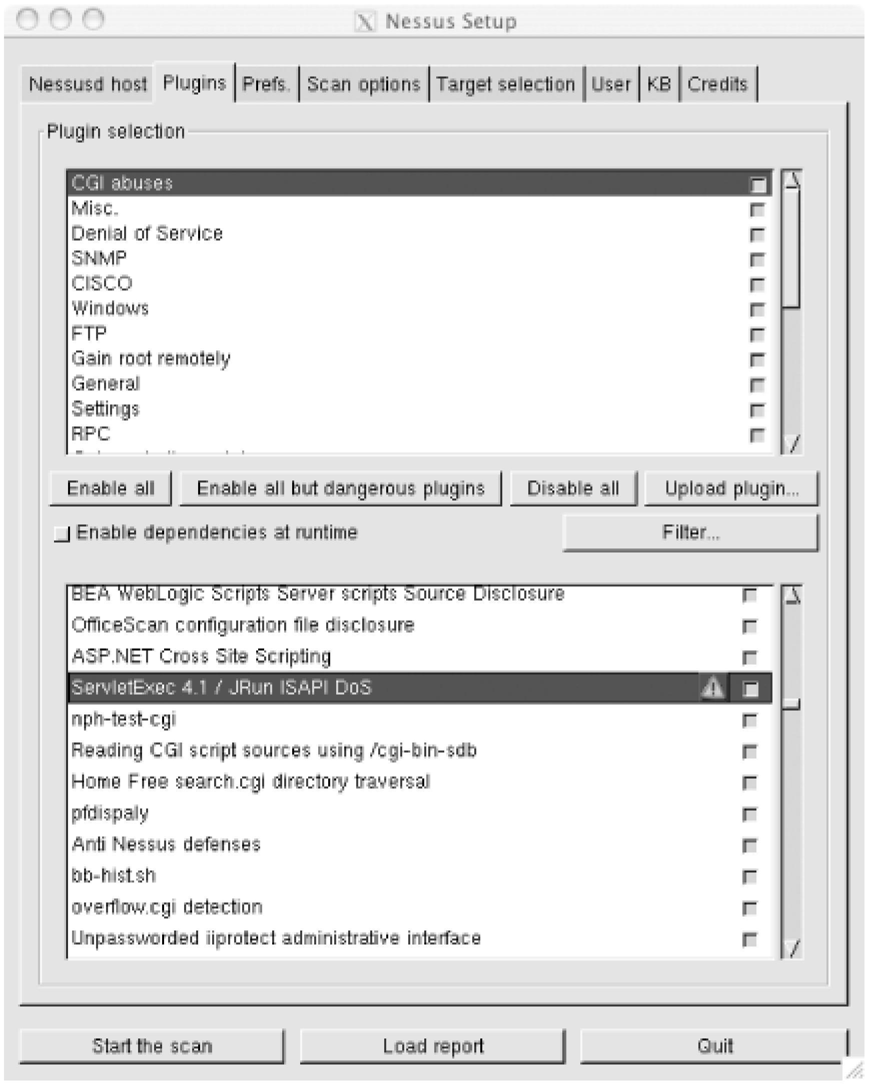
Login : andrew

Password :

You'll need to fill in the information for the user that you created and click the "Log In" button. After that, you'll be presented with a dialog that allows you to verify the information contained in the server's certificate.

To select which types of vulnerabilities to scan for, click on the Plugins tab, and you'll see something similar to Figure 3-9.

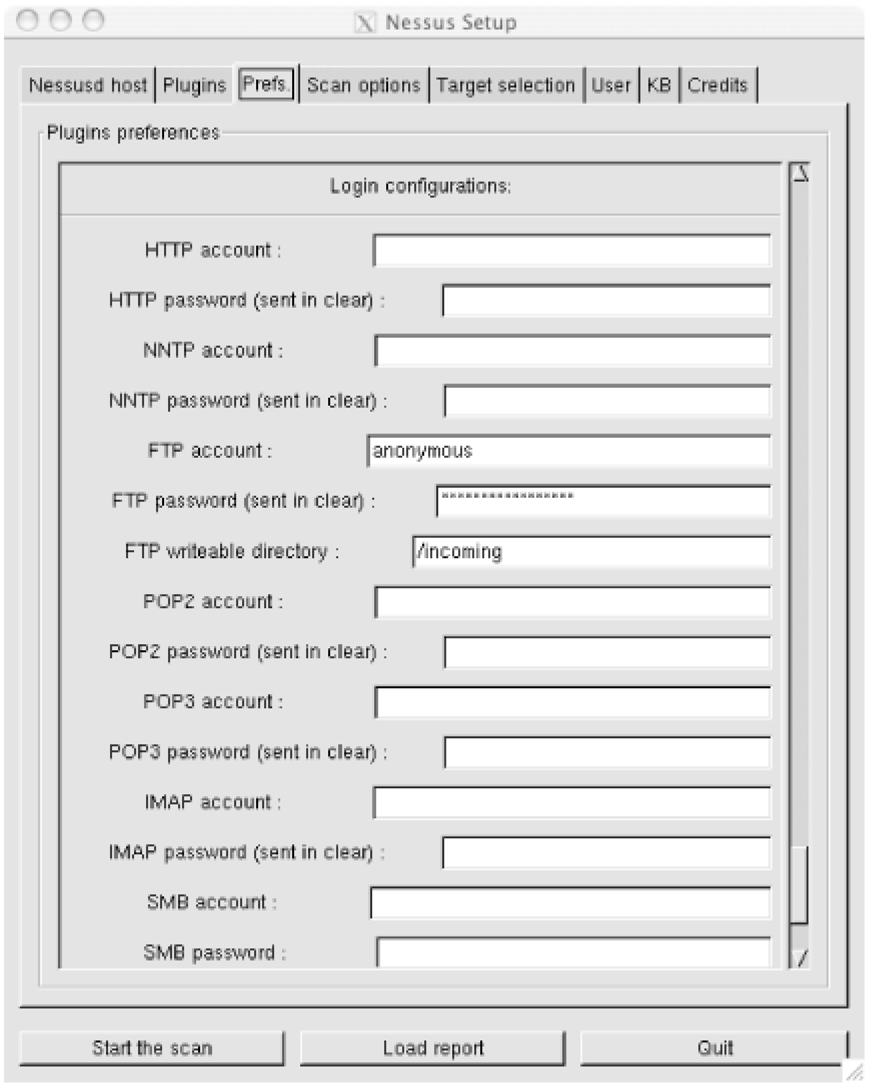
Figure 3-9. Nessus plugin selection


Nessus Setup
Nessusd host
Plugins
Prefs.
Scan options
Target selection
User
KB
Credits
Plugin selection
CGI abuses
Misc.
Denial of Service
SNMP
CISCO
Windows
FTP
Gain root remotely
General
Settings
RPC
Enable all
Enable all but dangerous plugins
Disable all
Upload plugin...
Enable dependencies at runtime
Filter...
BEA WebLogic Scripts Server scripts Source Disclosure
OfficeScan configuration file disclosure
ASP.NET Cross Site Scripting
ServletExec 4.1 / JRun ISAPI DoS
nph-test-cgi
Reading CGI script sources using /cgi-bin-sdb
Home Free search.cgi directory traversal
pfdispaly
Anti Nessus defenses
bb-hist.sh
overflow.cgi detection
Unpassworded iiprotect administrative interface
Start the scan
Load report
Quit

In the top pane you can enable or disable types of scans, and in the bottom pane you can disable individual vulnerability checks that belong to the category selected in the top pane. One thing to note: scans listed in the bottom pane that have an exclamation icon next to them will potentially crash the server that they are run against. If you want to enable all scans except for these, you can click the "Enable all but dangerous plugins" button. If you're running *Nessus* on a noncritical machine, you can probably leave these scans on, but you have been warned! You'll probably want to disable several types of scans unless you need to scan a machine or group of machines that run a wide variety of services; otherwise, you'll waste time having *Nessus* scan for services that you aren't running. For instance, if you wanted to scan a Solaris system, you might disable CGI abuses, CISCO, Windows, Peer-To-Peer File Sharing, Backdoors, Firewalls, Windows User Management, and Netware plug-ins.

In order for *Nessus* to more thoroughly test your services, you can supply it with login information for various services. This way, it can actually log into the service that it is testing and have access just like any normal user. You can tell *Nessus* about the accounts to use with the Prefs tab, as shown in Figure 3-10.

Figure 3-10. Nessus's Prefs tab



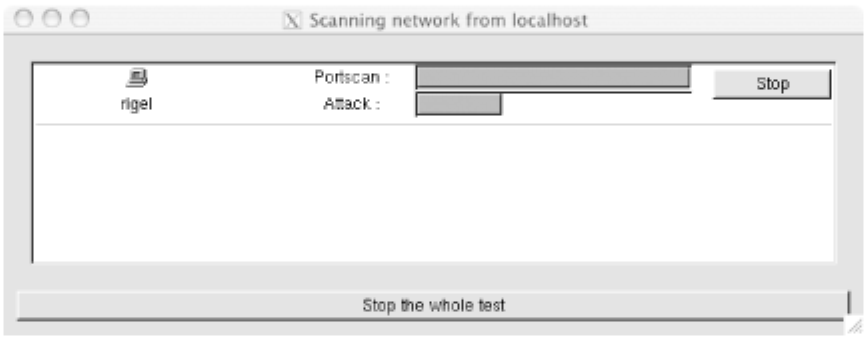
HTTP account :	
HTTP password (sent in clear) :	
NNTP account :	
NNTP password (sent in clear) :	
FTP account :	anonymous
FTP password (sent in clear) :	******************
FTP writeable directory :	/incoming
POP2 account :	
POP2 password (sent in clear) :	
POP3 account :	
POP3 password (sent in clear) :	
IMAP account :	
IMAP password (sent in clear) :	
SMB account :	
SMB password :	

In addition, you can tell *Nessus* to attempt brute-force logins to the services it is scanning. This can be a good testnot only of the services themselves, but also of your intrusion detection system (IDS) [Hack #82] and your system logs.

The "Scan options" tab lets you configure how *Nessus* will conduct its port-scans. Most of these settings can be left at their default value, unless you are also checking to see whether *Nessus* can evade detection by the hosts that you are scanning. For instance, *Nessus* is configured by default to perform full TCP connect scans and to ping the remote host that it is scanning. You can change this behavior by going to the "Scan options" tab, enabling "SYN scans" instead of "TCP connect", and disabling the ping. To specify which hosts you want to scan, you can use the "Target selection" tab.

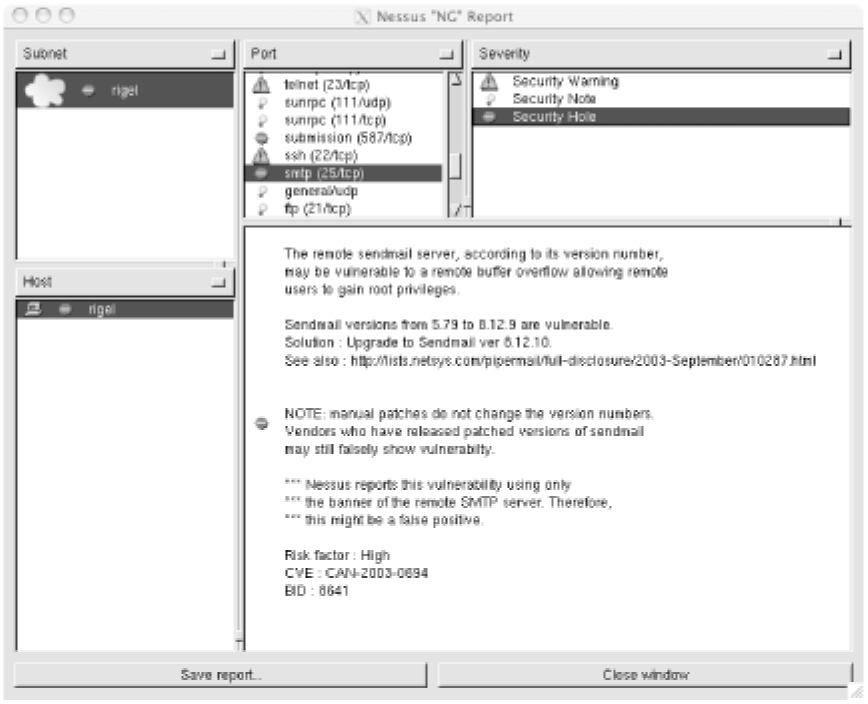
After you've made your selections, try scanning a host by clicking "Start the scan" at the bottom of the window. You should now see a window similar to Figure 3-11. In this case, *Nessus* is performing a scan against a Solaris machine.

Figure 3-11. Performing a vulnerability scan



The results of the scan are shown in Figure 3-12.

Figure 3-12. The vulnerability scan results



```
The remote sendmail server, according to its version number,
may be vulnerable to a remote buffer overflow allowing remote
users to gain root privileges.

Sendmail versions from 5.79 to 8.12.9 are vulnerable.
Solution : Upgrade to Sendmail ver 8.12.10.
See also : http://lists.netsys.com/pipermail/full-disclosure/2003-September/010287.html


NOTE: manual patches do not change the version numbers.
Vendors who have released patched versions of sendmail
may still falsely show vulnerabilty.

*** Nessus reports this vulnerability using only
*** the banner of the remote SMTP server. Therefore,
*** this might be a false positive.

Risk factor : High
CVE : CAN-2003-0694
BID : 8641
```

If you scanned multiple subnets, you can select those in the Subnet pane. Any hosts that are in the selected subnet will then appear in the Host pane. Similarly, when you select a host, the list of open ports on it will appear in the Port pane. You can select these to view the warnings, notes, and possible security holes that were found regarding the selected port. You can view the information that *Nessus* provides for these by clicking on them in the Severity pane. Don't be too alarmed by most of *Nessus*'s security notes and warnings; they are designed mainly to let you know what services you are running and to tell you if that service might present a potential vulnerability. Security holes are far more serious and should be investigated.

To save the report that you are viewing, click the "Save report" button. *Nessus* will let you save reports in a variety of formats. If you want to view the report in *Nessus* again at a later date, you should use *Nessus*'s own report format (NBE). Reports in this format can be viewed by using the "Load report" button in the main *Nessus* client window. Additionally, you can save reports in XML, HTML, ASCII, and even LaTeX format.

While Nmap is probably the champion of host and port detection, Nessus goes even further to demonstrate whether your own services are vulnerable to known attacks. Of course, new exploits surface all of the time, so it is important to keep your Nessus plug-ins up-to-date. Using Nessus, you can protect your own services by attempting to break into them before the bad boys do.

# US CO ntp1.linuxmedialabs.com Location: Linux Media Labs LLC, Colorado Springs, CO

Service Area: US

Synchronization: NTP Secondary (stratum 2), i686/Linux Access Policy: open access Contact: ntp@linuxmedialabs.com Note: ntp1 is an alias and the IP address may change, please use DNS

# US CO ntp1.tummy.com

Location: tummy.com, ltd., Fort Collins, CO

Service Area: US

Synchronization: NTP Secondary (stratum 2), i686/Linux Access Policy: open access.

Contact: ntp@tummy.com

Note: ntp1 is an alias and the IP address may change, please use DNS.

server ntp1.linuxmedialabs.com server ntp1.tummy.com

driftfile /etc/ntp.drift

Of course, if you're keeping all of your `ntpd` configuration files in */etc/ntp*, you'll want to use a directory similar to */etc/ntp/ntp.drift* instead.

That's it. Simply add `ntpd` to your startup scripts, start it up, and you're ready to go.

```
$<b>./CA.pl -newca</b>

CA certificate filename (or enter to create) Making CA certificate ...

Generating a 1024 bit RSA private key

..........++++++

.....................++++++

writing new private key to './demoCA/private/cakey.pem'

Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:

-----

You are about to be asked to enter information that will be incorporated into
your certificate request.

What you are about to enter is what is called a Distinguished Name or a
DN.

There are quite a few fields but you can leave some blank For some fields
there will be a default value, If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) []:<b>US</b> State or Province Name (full
name) []:<b>Colorado</b> Locality Name (eg, city) []:<b>Denver</b>
Organization Name (eg, company) []:<b>NonExistant Enterprises</b>
Organizational Unit Name (eg, section) []:<b>IT Services</b> Common
Name (eg, fully qualified host name) []:<b>ca.nonexistantdomain.com</b>
Email Address []:<b>certadmin@nonexistantdomain.com</b>

$ <b>ls -l demoCA/</b>
```

```
total 16

-rw-r--r-- 1 andrew andrew 1399 3 Dec 19:52 cacert.pem drwxr-xr-x 2
andrew andrew 68 3 Dec 19:49 certs drwxr-xr-x 2 andrew andrew 68 3 Dec
19:49 crl -rw-r--r-- 1 andrew andrew 0 3 Dec 19:49 index.txt drwxr-xr-x 2
andrew andrew 68 3 Dec 19:49 newcerts drwxr-xr-x 3 andrew andrew 102
3 Dec 19:49 private -rw-r--r-- 1 andrew andrew 3 3 Dec 19:49 serial

$DAYS="-days 365";

$ <b>openssl req -new -x509 -keyout cakey.pem -out \</b> <b>cakey.pem -
days 3650</b>

$ <b>./CA.pl -newreq-nodes</b> Generating a 1024 bit RSA private key

...++++++

..............................................++++++

writing new private key to 'newreq.pem'

-----

You are about to be asked to enter information that will be incorporated into
your certificate request.

What you are about to enter is what is called a Distinguished Name or a
DN.

There are quite a few fields but you can leave some blank For some fields
there will be a default value, If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:<b>US</b> State or Province Name
(full name) [Some-State]:<b>Colorado</b> Locality Name (eg, city) []:
<b>Denver</b> Organization Name (eg, company) [Internet Widgits Pty
Ltd]:<b>NonExistant Enterprises</b> Organizational Unit Name (eg,
section) []:<b>IT Services</b> Common Name (eg, YOUR name) []:
```

```
<b>mail.nonexistantdomain.com</b> Email Address []:
<b>postmaster@nonexistantdomain.com</b> Please enter the following
'extra' attributes to be sent with your certificate request A challenge
password []:

An optional company name []:NonExistant Enterprises Request (and
private key) is in newreq.pem

$ <b>./CA.pl -sign</b>

Using configuration from /System/Library/OpenSSL/openssl.cnf Enter pass
phrase for ./demoCA/private/cakey.pem: Check that the request matches the
signature Signature ok

Certificate Details:

   Serial Number: 1 (0x1) Validity

   Not Before: Dec 3 09:05:08 2003 GMT

   Not After : Dec 3 09:05:08 2004 GMT

   Subject:

   countryName = US

   stateOrProvinceName = Colorado localityName = Denver
organizationName = NonExistant Enterprises organizationalUnitName = IT
Services commonName = mail.nonexistantdomain.com emailAddress =
postmaster@nonexistantdomain.com X509v3 extensions: X509v3 Basic
Constraints: CA:FALSE

   Netscape Comment: OpenSSL Generated Certificate X509v3 Subject
Key Identifier:
94:0F:E9:F5:22:40:2C:71:D0:A7:5C:65:02:3E:BC:D8:DB:10:BD:88

   X509v3 Authority Key Identifier:
keyid:7E:AF:2D:A4:39:37:F5:36:AE:71:2E:09:0E:49:23:70:61:28:5F:4A
```

```
DirName:/C=US/ST=Colorado/L=Denver/O=NonExistant
Enterprises/OU=IT Services/

CN=Certificate
Administration/emailAddress=certadmin@nonexistantdomain.com
serial:00

Certificate is to be certified until Dec 7 09:05:08 2004 GMT (365 days)
Sign the certificate? [y/n]:<b>y</b> 1 out of 1 certificate requests certified,
commit? [y/n]<b>y</b> Write out database with 1 new entries
```

Data Base Updated

Signed certificate is in newcert.pem

Now you can set up keys in this manner for each server that needs to provide an SSL-encrypted service. It is easier to do this if you designate a single workstation to maintain the certificate authority and all the files associated with it. Don't forget to distribute your CA cert to programs that need to trust it [Hack #46] .

Hack 46 Distribute Your CA to Clients

Be sure all of your clients trust your new Certificate Authority.

Once you have created a Certificate Authority (CA) [Hack #45], any certificates that are signed by your CA will be trusted by any program that trusts your CA. To establish this trust, you need to distribute your CA's certificate to each program that needs to trust it. This could include email programs, IPSec installations, or web browsers.

Since SSL uses public key cryptography, there is no need to keep the certificate a secret. You can simply install it on a web server and download it to your clients over plain old HTTP. While the instructions for installing a CA cert are different for every program, this hack will show you a quick and easy way to install your CA on web browsers.

There are two possible formats that browsers will accept for new CA certs: *pem* and *der*. You can generate a *der* from your existing *pem* with a single `openssl` command:

```
$ openssl x509 -in demoCA/cacert.pem -outform DER -out cacert.der
```

Also, add the following line to the *conf/mime.types* file in your Apache installation:

```
application/x-x509-ca-cert      der pem crt
```

Now restart Apache for the change to take effect. You should now be able to place both the *cacert.der* and *demoCA/cacert.pem* files anywhere on your web server and have clients install the new cert by simply clicking on either link.

Early versions of Netscape expected *pem* format, but recent versions will accept either. Internet Explorer is just the opposite (early IE would accept only *der* format, but recent versions will take both). Other browsers will generally accept either format.

You will get a dialog box in your browser when downloading the new Certificate Authority, asking if you'd like to continue. Accept the certificate, and that's all there is to it. Now SSL certs that are signed by your CA will be accepted without warning the user.

Keep in mind that Certificate Authorities aren't to be taken lightly. If you accept a new CA in your browser, you had better trust it completelya mischievous CA manager could sign all sorts of certs that you should never trust, but your browser would never complain (since you claimed to trust the CA when you imported it). Be very careful about who you extend your trust to when using SSL-enabled browsers. It's worth looking around in the CA cache that ships with your browser to see exactly who you trust by default.

For example, did you know that AOL/Time Warner has its own CA? How about GTE? Or VISA? CA certs for all of these entities (and many others) ship with Netscape 7.0 for Linux, and are all trusted authorities for web sites, email, and application add-ons by default. Keep this in mind when browsing to SSL-enabled sites: if any of the default authorities have signed online content, your browser will trust it without requiring operator acknowledgment.

If you value your browser's security (and, by extension, the security of your client machine), then make it a point to review your trusted CA relationships.

Rob Flickenger (Linux Server Hacks)

```
$ <b>tar xfz imap.tar.Z</b>


$ <b>cd imap-2002e</b>


$ <b>make lnp SSLDIR=/usr SSLCERTS=/usr/share/ssl/certs</b>
In file included from /usr/include/openssl/ssl.h:179,
    from osdep.c:218:


/usr/include/openssl/kssl.h:72:18: krb5.h: No such file or directory
In file included from /usr/include/openssl/ssl.h:179,
    from osdep.c:218:
$ make lnp SSLDIR=/usr SSLCERTS=/usr/share/ssl/certs \
EXTRACFLAGS=-I/usr/kerberos/include
# <b>cp imapd/imapd ipopd/ipopd /usr/local/bin</b>
$ <b>openssl req -new -x509 -nodes \</b>
<b> -out /usr/share/ssl/certs/imapd.pem \</b>
<b> -keyout /usr/share/ssl/certs/imapd.pem -days 3650</b>
$ <b>openssl req -new -x509 -nodes \</b>
<b> -out /usr/share/ssl/certs/ipopd.pem \</b>
<b> -keyout /usr/share/ssl/certs/ipopd.pem -days 3650</b>
```

```
imaps stream tcp nowait root /usr/libexec/tcpd /usr/local/bin/imapd

pop3s stream tcp nowait root /usr/libexec/tcpd /usr/local/bin/ipop3d

# <b>kill -HUP `ps -ax | grep inetd | grep -v grep \</b>

    <b>| awk '{print $1}'`</b>
```

That's the final task for the server end of things. All you need to do now is configure your email clients to connect to the secure version of the service that you were using. Usually, there will be a Use Encryption, Use SSL, or some other similarly named checkbox in the incoming mail settings for your client. Just check the box, reconnect, and you should be using SSL now. Be sure your client trusts your CA cert, or you will be nagged with annoying (but important!) trust warnings.

```
$ <b>sendmail -bt -d0.1</b>
```

```
APPENDDEF(`conf_sendmail_ENVDEF', `-DSTARTTLS')
APPENDDEF(`conf_sendmail_LIBS', `-lssl -lcrypto')
```

```
# <b>./Build -c && ./Build install</b>
```

```
define(`confCACERT_PATH', `/etc/mail/certs') define(`confCACERT',
`/etc/mail/certs/cacert.pem') define(`confSERVER_CERT',
`/etc/mail/certs/cert.pem') define(`confSERVER_KEY',
`/etc/mail/certs/key.pem') define(`confCLIENT_CERT',
`/etc/mail/certs/cert.pem') define(`confCLIENT_KEY',
`/etc/mail/certs/key.pem')
```

```
# <b>telnet localhost smtp</b> Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

220 mail.example.com ESMTP Sendmail 8.12.9/8.12.9; Sun, 11 Jan 2004
12:07:43 -0800 (PST) <b>ehlo localhost</b> 250-mail.example.com Hello
IDENT:6l4ZhaGP3Qczqknqm/KdTFGsrBe2SCYC@localhost [127.0.0.1],
pleased to meet you 250-ENHANCEDSTATUSCODES

250-PIPELINING

250-EXPN

250-VERB

250-8BITMIME

250-SIZE

250-DSN

250-ETRN
```

```
250-AUTH DIGEST-MD5 CRAM-MD5

250-STARTTLS

250-DELIVERBY
```

250 HELP

<b>QUIT</b>

221 2.0.0 mail.example.com closing connection Connection closed by foreign host.

When `sendmail` relays mail to another TLS-enabled mail server, your mail will be encrypted. Now all you need to do is configure your mail client to use TLS when connecting to your mail server, and your users' email will be protected all the way to the MTA.

While there isn't enough room in this hack to cover every MTA available, nearly all support some variant of TLS. If you are running Exim (http://www.exim.org) or Courier (http://www.courier-mta.org), you can build TLS support straight out of the box. Postfix (http://www.postfix.org) has TLS support and is designed to be used in conjunction with Cyrus-SASL (see the HOWTO at http://postfix.state-of-mind.de/patrick.koetter/smtpauth/). Qmail has an RFC 2487 (TLS) patch available at http://inoa.net/qmail-tls/. With TLS support in virtually all MTAs and email clients, there is no longer any good reason to send email "in the clear."

```
# <b>/sbin/ifconfig eth1</b> eth1 Link encap:Ethernet HWaddr
00:E0:81:03:D8:8F

   BROADCAST MULTICAST MTU:1500 Metric:1

   RX packets:0 errors:0 dropped:0 overruns:0 frame:0

   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

   collisions:0 txqueuelen:100

   RX bytes:0 (0.0 b) TX bytes:0 (0.0 b) Interrupt:11 Base address:0x1c80

# <b>/sbin/ifconfig eth0 hw ether 00:DE:AD:BE:EF:00</b> #
<b>/sbin/ifconfig eth1</b> eth1 Link encap:Ethernet HWaddr
00:DE:AD:BE:EF:00

   BROADCAST MULTICAST MTU:1500 Metric:1

   RX packets:0 errors:0 dropped:0 overruns:0 frame:0

   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

   collisions:0 txqueuelen:100

   RX bytes:0 (0.0 b) TX bytes:0 (0.0 b) Interrupt:11 Base address:0x1c80

$ <b>./configure && make</b>

colossus # <b>sniffdet -i eth0 -t arp sirius</b> ----------------------------------
-------------------------
```

```
Sniffdet Report

Generated on: Wed Dec 31 03:49:28 2003

------------------------------------------------------------
```

```
Tests Results for target sirius

------------------------------------------------------------

Test: ARP Test (single host)

   Check if target replies a bogus ARP request (with wrong MAC)
Validation: OK

Started on: Wed Dec 31 03:49:08 2003

Finished on: Wed Dec 31 03:49:28 2003

Bytes Sent: 252

Bytes Received: 0

Packets Sent: 6

Packets Received: 0

------------------------------------------------------------

RESULT: NEGATIVE

------------------------------------------------------------

------------------------------------------------------------

Number of valid tests: #1

Number of tests with positive result: #0

------------------------------------------------------------
```

```
sirius # <b>tcpdump -i le0 arp</b> tcpdump: listening on le0

06:58:00.458836 arp who-has sirius.nnc tell colossus.nnc 06:58:00.458952
arp reply sirius.nnc is-at 8:0:20:81:a4:a3
```

```
06:58:00.466601 arp who-has sirius.nnc (ff:0:0:0:0:0) tell colossus.nnc
06:58:00.466928 arp reply sirius.nnc is-at 8:0:20:81:a4:a3
```

```
Sniffdet Report

Generated on: Wed Dec 31 06:58:01 2003

------------------------------------------------------------
```

```
Tests Results for target sirius

------------------------------------------------------------

Test: ARP Test (single host)

   Check if target replies a bogus ARP request (with wrong MAC)
Validation: OK

Started on: Wed Dec 31 06:58:00 2003

Finished on: Wed Dec 31 06:58:01 2003

Bytes Sent: 84

Bytes Received: 60

Packets Sent: 2

Packets Received: 1

------------------------------------------------------------

RESULT: POSITIVE

------------------------------------------------------------

------------------------------------------------------------

Number of valid tests: #1

Number of tests with positive result: #1

------------------------------------------------------------
```

The DNS test works very well, particularly on shared-medium networks such as hubs or wireless LANs. However, it does rely on name resolution being enabled in the sniffer. When performing DNS tests, *sniffdet* will send bogus packets that contain IP addresses that are not in use on the local network segment. If name resolution is enabled, the sniffer will attempt to do a reverse lookup in order to determine the hostname that corresponds to the IP addresses. Since these addresses are not in use, *sniffdet* will determine that the target machine is in promiscuous mode when it sees the DNS queries.

This test can be performed just as the ARP test, but instead of using `-t arp`, use `-t dns`.

```
# <b>./configure \ </b>
```

```
<span class="docEmphBold">--with-apache=../apache_1.3.29 \</span>
<span class="docEmphBold">--with-ssl=SYSTEM \</span> <span
class="docEmphBold">--prefix=/usr/local/apache \</span> <span
class="docEmphBold">--enable-module=most \</span> <span
class="docEmphBold">--enable-module=mmap_static \</span> <span
class="docEmphBold">--enable-module=so \</span> <span
class="docEmphBold">--enable-shared=ssl \</span> <span
class="docEmphBold">--disable-rule=SSL_COMPAT \</span> <span
class="docEmphBold">--server-uid=www \</span> <span
class="docEmphBold">--server-gid=www \</span> <span
class="docEmphBold">--enable-suexec \</span> <span
class="docEmphBold">--suexec-caller=www \</span> <span
class="docEmphBold">--suexec-uidmin=500 \</span> <span
class="docEmphBold">--suexec-gidmin=500</span>
```

```
# <b>/usr/local/apache/bin/apachectl startssl</b>
```

```
# <b>/usr/local/apache/bin/apacectl start</b>
```

```
# <b>grep suexec /usr/local/apache/logs/error_log</b> [Thu Jan 1 16:48:17
2004] [notice] suEXEC mechanism enabled (wrapper:
/usr/local/apache/bin/suexec)
```

```
<Directory /home/*/public_html>

   AllowOverride FileInfo AuthConfig Limit Options MultiViews Indexes
SymLinksIfOwnerMatch Includes ExecCGI <Limit GET POST OPTIONS
PROPFIND> Order allow,deny Allow from all

   </Limit>

   <LimitExcept GET POST OPTIONS PROPFIND> Order deny,allow
Deny from all

   </LimitExcept> </Directory>
```

```
AddHandler cgi-script .cgi

# <b>/usr/local/apache/bin/apachectl restart</b>

#!/bin/sh

echo -e "Content-Type: text/plain\r\n\r\n"

/usr/sbin/id

uid=80(www) gid=80(www) groups=80(www)

$ <b>mkdir public_html && chmod 711 ~/ ~/public_html</b> $ <b>cp
/usr/local/apache/cgi-bin/suexec-test.cgi .</b>

uid=500(andrew) gid=500(andrew) groups=500(andrew)

<VirtualHost>

   User myuser

   Group mygroup

   DocumentRoot /usr/local/apache/htdocs/mysite ...

</VirtualHost>
```

Unfortunately, suEXEC is incompatible with `mod_perl` and `mod_php` because the modules run within the Apache process itself instead of a separate program. Since the Apache process is running as a nonroot user it cannot change the UID under which the scripts execute. suEXEC works by having Apache call a special SUID wrapper (e.g., */usr/local/apache/bin/suexec*) that can only be invoked by Apache processes. If you care to make the security/performance trade-off by using suEXEC but still need to run Perl scripts, you can do so through the standard CGI interface. Just as with Perl, you can also run PHP programs through the CGI interface, but you'll have to create a `php` binary and specify it as the interpreter in all the PHP scripts you wish to execute through suEXEC. You can also execute your scripts through `mod_perl` or `mod_php` by locating them outside the directories where suEXEC will work.

```
# <b>mkdir /named_chroot</b>

# <b>cd /named_chroot</b>


# <b>mkdir -p dev etc/namedb/slave var/run</b>

# <b>cp /etc/named.conf /named_chroot/etc</b>

# <b>cp -a /var/namedb/* /named_chroot/etc/namedb</b>

# <b>chown -R named:named /named_chroot/etc/namedb/slave</b>

# <b>chown named:named /named_chroot/var/run</b>

# <b>cd /named_chroot/dev</b>

# <b>ls -la /dev/null /dev/random</b>

crw-rw-rw- 1 root root 1, 3 Jan 30 2003 /dev/null

crw-r--r-- 1 root root 1, 8 Jan 30 2003 /dev/random

# <b>mknod null c 1 3</b>


# <b>mknod random c 1 8</b>

# <b>chmod 666 null random</b>

SYSLOGD_OPTIONS="-m 0 -a /named_chroot/dev/log"

syslogd_flags="-s -a /named_chroot/dev/log"

# <b>named -u named -t /named_chroot</b>

allow-transfer {
    192.168.1.20;

    192.168.1.21;

};

version "SuperHappy DNS v1.5";
```

Note that this really doesn't provide extra security, but if you don't want to advertise what software and version you're running to the entire world, you don't have to.

See Also

- The section "Securing BIND" in Building Secure Servers with Linux, by Michael D. Bauer (O'Reilly)

```
$ <b>./configure --prefix=/mysql --with-mysqld-ldflags=-all-static &&
make</b>

# <b>make install DESTDIR=/mysql_chroot && ln -s
/mysql_chroot/mysql /mysql</b> # <b>scripts/mysql_install_db</b>

# <b>chown -R root:mysql /mysql</b> # <b>chown -R mysql
/mysql/var</b>

# <b>/mysql/bin/mysqld_safe&</b> Starting mysqld daemon with
databases from /mysql/var # <b>ps -aux | grep mysql | grep -v grep</b>
root 10137 0.6 0.5 4156 744 pts/2 S 23:01 0:00 /bin/sh /mysql/bin/

mysqld_safe

mysql 10150 7.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10151 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10152 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10153 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10154 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10155 0.3 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10156 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10157 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10158 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

mysql 10159 0.0 9.3 46224 11756 pts/2 S 23:01 0:00 [mysqld]

# <b>/mysql/bin/mysqladmin shutdown</b>
```

040103 23:02:45 mysqld ended

```
[1]+ Done /mysql/bin/mysqld_safe

# <b>mkdir /mysql_chroot/tmp /mysql_chroot/dev</b> # <b>chmod 1777 /mysql_chroot/tmp</b> # <b>ls -l /dev/null</b>

crw-rw-rw- 1 root root 1, 3 Jan 30 2003 /dev/null # <b>mknod /mysql_chroot/dev/null c 1 3</b>

# <b>/usr/sbin/chroot /mysql_chroot /mysql/libexec/mysqld -u 100</b>
```

```
if test -z "$args"

   then

   $NOHUP_NICENESS $ledir/$MYSQLD $defaults \ --basedir=$MY_BASEDIR_VERSION \ --datadir=$DATADIR $USER_OPTION \ --pid-file=$pid_file --skip-locking >> $err_log 2>&1

   else

   eval "$NOHUP_NICENESS $ledir/$MYSQLD $defaults \ --basedir=$MY_BASEDIR_VERSION \ --datadir=$DATADIR $USER_OPTION \ --pid-file=$pid_file --skip-locking $args >> $err_log 2>&1"

   fi
```

```
if test -z "$args"

   then

   $NOHUP_NICENESS /usr/sbin/chroot /mysql_chroot \ $ledir/$MYSQLD $defaults \ --basedir=$MY_BASEDIR_VERSION \ --datadir=$DATADIR $USER_OPTION \ --pid-file=$pid_file --skip-locking >> $err_log 2>&1
   else

   eval "$NOHUP_NICENESS /usr/sbin/chroot /mysql_chroot \
$ledir/$MYSQLD $defaults \ --basedir=$MY_BASEDIR_VERSION \ --
datadir=$DATADIR $USER_OPTION \ --pid-file=$pid_file --skip-locking
$args >> $err_log 2>&1"

   fi

# <b>/mysql/bin/mysqld_safe --user=100</b>
```

In addition, you may want to create a separate *my.conf* file for the MySQL utilities and server. For instance, in */etc/my.cnf* you could specify `socket = /mysql_chroot/tmp/mysql.sock` in the `[client]` section so you do not have to manually specify the socket every time you run a MySQL-related program.

You'll also probably want to disable MySQL's ability to load data from local files. To do this, you can add `set-variable=local-infile=0` to the `[mysqld]` section of your */mysql_chroot/etc/my.cnf*. This disables MySQL's `LOAD DATA LOCAL INFILE` command. Alternatively, you can disable it from the command line by using the `--local-infile=0` option.

```
$ <b>./configure && make</b>

$ <b>./configure --with-sfsuser=nobody --with-sfsgroup=nobody</b>

# <b>sfscd</b>

# <b>cd /sfs/@sfs.fs.net,uzwadtctbjb3dg596waiyru8cx5kb4an</b> #
<b>ls</b>

CONGRATULATIONS cvs pi0 reddy sfswww

# <b>cat CONGRATULATIONS</b> You have set up a working SFS
client.
```

```
#
```

```
# <b>mkdir /etc/sfs</b>
```

```
# <b>sfskey gen -P /etc/sfs/sfs_host_key</b>
```

```
Export /var/sfs/root /
```

```
Export /home /home
```

```
/var/sfs/root localhost(rw)
```

```
/home localhost(rw)
```

```
Dec 12 12:29:14 colossus : sfssd: version 0.7.2, pid 3503
```

```
Dec 12 12:29:14 colossus : rexd: version 0.7.2, pid 3505
```

```
Dec 12 12:29:14 colossus : sfsauthd: version 0.7.2, pid 3506
```

```
Dec 12 12:29:14 colossus : rexd: serving
@colossus.nnc,fd82m36uwxj6m3q8tawp56ztgsvu7g77
```

```
Dec 12 12:29:14 colossus : rexd: spawning /usr/local/lib/sfs-0.7.2/ptyd Dec
12 12:29:15 colossus rpc.mountd: authenticated mount request from
localhost.localdomain:715 for /var/sfs/root (/var/sfs/root) Dec 12 12:29:15
colossus rpc.mountd: authenticated mount request from
localhost.localdomain:715 for /home (/home) Dec 12 12:29:15 colossus :
sfsauthd: serving @colossus.nnc,fd82m36uwxj6m3q8tawp56ztgsvu7g77
```

```
Dec 12 12:29:16 colossus : sfsrwsd: version 0.7.2, pid 3507
```

```
Dec 12 12:29:16 colossus : sfsrwsd: serving
/sfs/@colossus.nnc,fd82m36uwxj6m3q8tawp56ztgsvu7g77
```

```
$ <b>sfskey register</b>
```

```
sfskey: /home/andrew/.sfs/random_seed: No such file or directory sfskey:
creating directory /home/andrew/.sfs sfskey: creating directory
/home/andrew/.sfs/authkeys /var/sfs/sockets/agent.sock: No such file or
directory sfskey: sfscd not running, limiting sources of entropy Creating
new key: andrew@colossus.nnc#1 (Rabin) Key Label:
andrew@colossus.nnc#1

Enter passphrase:

   Again:

sfskey needs secret bits with which to seed the random number generator.

Please type some random or unguessable text until you hear a beep:
```

DONE

UNIX password:

```
colossus.nnc: New SRP key: andrew@colossus.nnc/1024

wrote key: /home/andrew/.sfs/authkeys/andrew@colossus.nnc#1

$ <b>sfskey login andrew@colossus.nnc</b> Passphrase for
andrew@colossus.nnc/1024: SFS Login as andrew@colossus.nnc

$ <b>cd /sfs/@colossus.nnc,fd82m36uwxj6m3q8tawp56ztgsvu7g77</b> $
<b>ls</b>

home
```

As you can see, SFS is a very powerful tool for sharing files across a network, and even across the Internet. Not only does it provide security, but it also provides a unique and universal method for referencing a remote host and its exported filesystems. You can even put your home directory on an SFS server, simply by linking the universal pathname of the exported filesystem */home*.

Chapter 4. Logging

Hacks #54-60

Keeping logs is a very important aspect of maintaining the security of your network, as logs can assist in everything from alerting you to an impending attack to debugging network problems.

After an incident has occurred, good logs can help you track down how the attacker got in, fix the security hole, and figure out which machines were affected. In addition, logs can help with tracing the attack back to its source, so you can identify or take legal action against the intruder.

In short, log files are worth their weight in gold (just pretend that bits and bytes weigh a lot).

As such, they should be given at least as much protection as any other information that's stored on your

serverseven the patent schematics for your perpetual motion machine.

This chapter deals mostly with various ways to set up remote logging, whether it be a simple central *syslogd* that your servers are logging to, setting up your Windows machines to send to a *syslogd*, or using *syslog-ng* to collect logs from remote sites through an encrypted TCP

connection. Using these methods, you

can ensure that your logs are sitting safely on a dedicated server that's running minimal services, to decrease the chance that the logs will be compromised.

Once you have all your logs collected in a central place, what can you do with them? This chapter also

covers ways to summarize your logs into reports that are easy to read and understand, so you can quickly spot the most pertinent information. If

that's not fast enough for you,

you'll also learn how to set up real-time alerts that will notify you as soon as a critical event occurs. In some circumstances, responding immediately to an eventrather than waiting around for it to end up in a report that you read the next morningcan save hours of effort.

```
# <b>/usr/sbin/syslogd -m 0 -r</b>
```

```
# <b>/usr/sbin/syslogd</b>
```

```
<tt><i>ipaddr</i></tt>/<tt><i>mask</i></tt>[:<tt><i>service</i></tt>]



   <tt><i>domain</i></tt>[:<tt><i>service</i></tt>]



   *<tt><i>domain</i></tt>[:<tt><i>service</i></tt>]
```

```
# <b>/usr/sbin/syslogd -a /var/empty/dev/log -u</b>
```

```
# <b>/usr/sbin/syslogd -T</b>
```

```
*.* @loghost
```

You can either make this the only line in the configuration file, in which case messages will be logged only to the remote host, or add it to what is already there, in which case logs will be stored both locally and remotely for safekeeping.

One drawback to remote logging is that the stock syslogd for most operating systems fails to provide any measure of authentication or access control with regard to who may write to a central log host. Firewalls can provide some protection, keeping out everyone but those who are determined to undermine your logging infrastructure; however, someone who has already gained access to your local network can easily spoof his network connection and bypass any firewall rules that you set up. If you've determined that this is a concern for your network, take a look at [Hack #59], which discusses one method for setting up remote logging using public-key authentication and SSL-encrypted connections.

```
auth.warning /var/log/auth

mail.err /var/log/maillog

kern.* /var/log/kernel


cron.crit /var/log/cron


*.err;mail.none /var/log/syslog

*.info;auth.none;mail.none /var/log/messages

#*.=debug /var/log/debug

local0.info /var/log/cluster

local1.err /var/log/spamerica

# <b>mkfifo -m 0664 /var/log/debug</b>

*.=debug |/var/log/debug

*.* |/var/log/monitor

Dec 29 18:33:35 catlin -- MARK --


Dec 29 18:53:35 catlin -- MARK --


Dec 29 19:13:35 catlin -- MARK --
Dec 29 19:33:35 catlin -- MARK --

Dec 29 19:53:35 catlin -- MARK --

Dec 29 20:13:35 catlin -- MARK --

Dec 29 20:33:35 catlin -- MARK --

Dec 29 20:53:35 catlin -- MARK --

Dec 29 21:13:35 catlin -- MARK --
```

```
# <b>killall syslogd; /usr/sbin/syslogd -m 0</b>
```

If all of this fiddling about with facilities and priorities strikes you as arcane Unix speak, you're not alone. These examples are provided for systems that include the default (and venerable) syslogd daemon. If you have the opportunity to install a new syslogd, you will likely want to look into syslog-ng. This new implementation of syslogd allows much more flexible filtering and a slew of new features. We take a look at some of what is possible with *syslog-ng* in [Hack #59] .

Rob Flickenger

```
C:\> <b>ntsyslog -install</b>
```

```
Oct 29 17:19:04 plunder security[failure] 529 NT AUTHORITY\\SYSTEM
Logon Failure:

Reason:Unknown user name or bad password User Name:andrew
Domain:PLUNDER Logon Type:2



Logon Process:User32 Authentication Package:Negotiate Workstation
Name:PLUNDER
```

One of the best things about doing this is that now you can use the wealth and flexibility of Unix log-monitoring tools to help monitor all your Windows systems.

```
# <b>tar xfz logwatch-5.0.tar.gz</b> # <b>cd logwatch-5.0</b>

# <b>mkdir /etc/log.d</b>

# <b>cp -R conf lib scripts /etc/log.d</b>

# <b>cp logwatch.8 /usr/share/man/man8</b> # <b>(cd /usr/sbin</b>
<b>&& \</b> <b>ln -s ../../etc/log.d/scripts/logwatch.pl logwatch)</b>

# <b>logwatch --print | less</b> ################### LogWatch 4.3.1
(01/13/03) ####################

   Processing Initiated: Sat Dec 27 21:12:26 2003

   Date Range Processed: yesterday Detail Level of Output: 0

   Logfiles for Host: colossus
```

```
################################################################################

-------------------- SSHD Begin ------------------------

Users logging in through sshd:

   andrew logged in from kryten.nnc (192.168.0.60) using password: 2
Time(s) ---------------------- SSHD End -------------------------

###################### LogWatch End #########################
```

```
LogFile = /var/log/myservice

Archive = /var/log/myservice.*

LogFile = myservice
```

Finally, since *logwatch* is merely a framework for generating log file summaries, you'll also need to create a script in */etc/log.d/scripts/services* called *myservice*. When *logwatch* executes, it will strip all time entries from the logs and pass the rest of the log entry through standard input to the *myservice* script. Therefore, you must write your script to read from standard input, parse out the pertinent information, and then print it to standard out.

This just scratches the surface of how to get *logwatch* running on your system. There is a great deal of information in the HOWTO-Make-Filter, which is included with the *logwatch* distribution.

```
# <b>perl Makefile.PL</b>


# <b>make && make install</b>
Warning: prerequisite Date::Calc 0 not found.


Warning: prerequisite Date::Parse 0 not found.


Warning: prerequisite Time::HiRes 1.12 not found.
```

```
Writing Makefile for swatch

# <b>perl -MCPAN -e "install Date::Calc"</b>

# <b>perl -MCPAN -e "install Date::Parse"</b>

# <b>perl -MCPAN -e "Time::HiRes"</b>

# <b>swatch -c /etc/swatch/messages.conf -t /var/log/messages</b>
```

```
watchfor /<<tt><i>regex</i></tt>>/


<<tt><i>action1</i></tt>>

[<tt><i>action2</i></tt>]


[<tt><i>action3</i></tt>]


...

ignore /<<tt><i>regex</i></tt>>/

echo blink,red

write <tt><i>user</i></tt>:<tt><i>user2</i></tt>:...

exec <<tt><i>command</i></tt>>

exec "<tt><i>mycommand</i></tt> $2 $3"

mail addresses=<tt><i>address</i></tt>:<tt><i>address2</i>
</tt>:...,subject=<tt><i>mysubject</i></tt>
```

pipe <tt><i>mycommand</i></tt>,keep_open

throttle <tt><i>h</i></tt>:<tt><i>m</i></tt>:<tt><i>s</i></tt>

The `throttle` action will throttle based on the contents of the message by default. If you would like to throttle the actions based on the regular expression that caused the match, you can add a `,use=regex` to the end of your `throttle` statement.

Swatch is an incredibly useful tool, but it can take some work to create a good *.swatchrc*. The best way to figure out what to look for is to examine your log files for behavior that you want to monitor closely.

```
$ <b>tar xfz libol-0.3.9.tar.gz</b> $ <b>cd libol-0.3.9</b>
```

```
$ <b>./configure && make</b>
```

```
$ <b>tar xfz syslog-ng-1.5.26.tar.gz</b> $ <b>cd syslog-ng-1.5.26</b> $
<b>./configure --with-libol=../libol-0.3.9</b> $ <b>make</b>
```

```
*.err;kern.debug;auth.notice;mail.crit /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages security.*
/var/log/security
```

```
auth.info;authpriv.info /var/log/auth.log mail.info /var/log/maillog
```

```
lpr.info /var/log/lpd-errs
```

```
cron.* /var/log/cron
```

```
*.emerg *
```

```
source src { unix-dgram("/var/run/log"); internal( ); };
```

```
source src { unix-stream("/dev/log"); internal( ) };
```

```
destination console { file("/dev/console"); }; destination messages {
file("/var/log/messages"); }; destination security { file("/var/log/security");
}; destination authlog { file("/var/log/auth.log"); }; destination maillog {
file("/var/log/maillog"); }; destination lpd-errs { file("/var/log/lpd-errs"); };
destination cron { file("/var/log/cron"); }; destination slip {
file("/var/log/slip.log"); }; destination ppp { file("/var/log/ppp.log"); };
destination allusers { usertty("*"); };
```

```
destination loghost { tcp("192.168.0.2" port(5140)); };
```

```
filter f_auth { facility(auth); };
```

```
filter f_authpriv { facility(authpriv); }; filter f_console { facility(console);
}; filter f_cron { facility(cron); };
```

```
filter f_daemon { facility(daemon); }; filter f_ftp { facility(ftp); };

filter f_kern { facility(kern); };

filter f_lpr { facility(lpr); };

filter f_mail { facility(mail); };

filter f_news { facility(news); };

filter f_security { facility(security); }; filter f_user { facility(user); };

filter f_uucp { facility(uucp); };

filter f_emerg { level(emerg); };

filter f_alert { level(alert..emerg); }; filter f_crit { level(crit..emerg); }; filter
f_err { level(err..emerg); };

filter f_warning { level(warning..emerg); }; filter f_notice {
level(notice..emerg); }; filter f_info { level(info..emerg); }; filter f_debug {
level(debug..emerg); };

# *.err;kern.debug;auth.notice;mail.crit /dev/console log { source(src);
filter(f_err); destination(console); }; log { source(src); filter(f_kern);
filter(f_debug); destination(console); }; log { source(src); filter(f_auth);
filter(f_notice); destination(console); }; log { source(src); filter(f_mail);
filter(f_crit); destination(console); }; #
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages log {
source(src); filter(f_notice); destination(messages); }; log { source(src);
filter(f_kern); filter(f_debug); destination(messages); }; log { source(src);
filter(f_lpr); filter(f_info); destination(messages); }; log { source(src);
filter(f_mail); filter(f_crit); destination(messages); }; log { source(src);
filter(f_news); filter(f_err); destination(messages); }; # security.*
/var/log/security

log { source(src); filter(f_security); destination(security); }; #
auth.info;authpriv.info /var/log/auth.log log { source(src); filter(f_auth);
filter(f_info); destination(authlog); }; log { source(src); filter(f_authpriv);
filter(f_info); destination(authlog); }; # mail.info /var/log/maillog

log { source(src); filter(f_mail); filter(f_info); destination(maillog); }; #
lpr.info /var/log/lpd-errs

log { source(src); filter(f_lpr); filter(f_info); destination(lpd-errs); }; #
cron.* /var/log/cron

log { source(src); filter(f_cron); destination(cron); }; # *.emerg *

log { source(src); filter(f_emerg); destination(allusers); };

source r_src { tcp(ip("192.168.0.2") port(5140)); };

source src {

   unix-dgram("/var/run/log"); tcp(ip("192.168.0.2") port(5140)); internal( );

};

destination r_all { file("/var/log/$HOST"); };

log { source(r_src); destination(r_all); };

destination fbsd_messages { file("/var/log/$HOST/messages"); };

options { create_dirs(yes); };

destination fbsd_messages {

   file("/var/log/$HOST/$YEAR.$MONTH.$DAY/messages"); };

log { source(r_src); destination(loghost); };
```

Since *syslog-ng* is now using TCP ports, you can use any encrypting tunnel you like to secure the traffic between your *syslog-ng* daemons. You can use SSH port forwarding [Hack #72] or stunnel [Hack #76] to create an encrypted channel between each of your servers. By limiting connections on the listening port to include only localhost (using firewall rules, as in [Hack #33] or [Hack #34]), you can eliminate the possibility of bogus log entries or denial-of-service attacks.

Server logs are among the most critical information that a system administrator needs to do her job. Using new tools and strong encryption, you can keep your valuable log data safe from prying eyes.

```
# <b>mkdir /var/account</b> # <b>touch /var/account/pacct && chmod
660 /var/account/pacct</b> # <b>/sbin/accton</b>
<b>/var/account/pacct</b>

# <b>chkconfig psacct on</b> # <b>/sbin/service psacct start</b>

# <b>chkconfig acct on</b> # <b>/sbin/service acct start</b>

[andrew@colossus andrew]$ <b>ac</b> total 106.23

# <b>ac -p</b>

   root 0.07

   andrew 106.05

   total 106.12

# <b>lastcomm andrew</b>

ls andrew ?? 0.01 secs Mon Dec 15 05:58

rpmq andrew ?? 0.08 secs Mon Dec 15 05:58

sh andrew ?? 0.03 secs Mon Dec 15 05:44

gunzip andrew ?? 0.00 secs Mon Dec 15 05:44

# <b>lastcomm bash</b>

bash F andrew ?? 0.00 secs Mon Dec 15 06:44

bash F root stdout 0.01 secs Mon Dec 15 05:20

bash F root stdout 0.00 secs Mon Dec 15 05:20

bash F andrew ?? 0.00 secs Mon Dec 15 05:19

# <b>sa</b>
```

```
   14 0.04re 0.03cp 0avio 1297k troff 7 0.03re 0.03cp 0avio 422k lastcomm
2 63.90re 0.01cp 0avio 983k info 14 34.02re 0.01cp 0avio 959k less 14
0.03re 0.01cp 0avio 1132k grotty 44 0.02re 0.01cp 0avio 432k gunzip

# <b>sa -u</b>

root 0.01 cpu 344k mem 0 io which

root 0.00 cpu 1094k mem 0 io bash

root 0.07 cpu 1434k mem 0 io rpmq

andrew 0.02 cpu 342k mem 0 io id

andrew 0.00 cpu 526k mem 0 io bash

andrew 0.01 cpu 526k mem 0 io bash

andrew 0.03 cpu 378k mem 0 io grep

andrew 0.01 cpu 354k mem 0 io id

andrew 0.01 cpu 526k mem 0 io bash

andrew 0.00 cpu 340k mem 0 io hostname
```

You can peruse the output of these commands every so often to look for suspicious activity, such as increases in CPU usage or commands that are known to be used for mischief.

Chapter 5. Monitoring and Trending

Hacks #61-66

While the importance of reliable system logs can't be overestimated, logs only tell part of the story of what is happening on your network. When

something out of the ordinary happens, the event is duly logged to the appropriate file, where it waits for a human to notice and take the appropriate action. But logs are

valuable only if someone actually reads them.

When log files add to the deluge of information that most network administrators already wade through each day, many log files may go unread for days or weeks.

This situation is made worse when log files are clogged with irrelevant information. For

example, a cry for help from an overburdened mail server can easily be lost if it is surrounded by innocuous entries about failed spam attempts. All too often, logs are

used as a resource to figure out "what

happened" when systems fail, rather than as a guide to what is happening now.

Another important aspect of log entries is that they only provide a "spot check" of your system at a

particular moment. Without a history

of what normal performance looks like, it can be difficult to tell the difference between ordinary network traffic, a DoS attack, and a visitation from Slashdot readers.

While you can easily build a report on how many times the */var* partition filled up, how can you easily know what usage looks like over time?

Is the mail spool clogged due to one inconsiderate user, or is it part of an attack by an adversary?

Or is it simply a general trend that is the result of trying to serve too many users on too small a disk?

This chapter describes a number of methods for tracking the availability of services and resources over time. Rather than having to watch system logs manually, it is usually far better to have the systems notify you when there is a problemand *only* when there is a problem. There are also a

number of suggestions about how to recognize trends in your network traffic by monitoring flows and plotting the results on a graph. Sure, you may know what your

average outbound Internet traffic looks like, but how much of that traffic is made up of HTTP versus SMTP? You may know roughly how much is being used by each server on your network, but what if you want to break the traffic down by protocol?

The hacks in this chapter will show you how.

```
$ <b>tar xfz nagios-1.1.tar.gz</b> $ <b>cd nagios-1.1</b>

$ <b>./configure</b> <b>--with-nagios-user=nagios --with-nagios-
grp=nagios</b>

$ <b>./configure --prefix=/usr/local/nagios \</b> <b>--with-nagios-
user=nagios --with-nagis-grp=nagios</b> $ <b>make all</b>

$ <b>/usr/local/nagios/libexec/check_ssh</b> check_ssh (nagios-plugins
1.4.0alpha1) 1.13

The nagios plugins come with ABSOLUTELY NO WARRANTY. You may
redistribute copies of the plugins under the terms of the GNU General
Public License.

For more information about these matters, see the file named COPYING.

Copyright (c) 1999 Remi Paulmier <remi@sinfomic.fr> Copyright (c)
2000-2003 Nagios Plugin Development Team <nagiosplug-
devel@lists.sourceforge.net> Try to connect to SSH server at specified
server and port Usage: check_ssh [-46] [-t <timeout>] [-p <port>] <host>
check_ssh (-h | --help) for detailed help check_ssh (-V | --version) for
version information Options:

-h, --help

   Print detailed help screen -V, --version

   Print version information -H, --hostname=ADDRESS

   Host name or IP Address -p, --port=INTEGER

   Port number (default: 22)
```

-4, --use-ipv4

Use IPv4 connection

-6, --use-ipv6

Use IPv6 connection

-t, --timeout=INTEGER

Seconds before connection times out (default: 10) -v, --verbose

Show details for command-line debugging (Nagios may truncate output) Send email to nagios-users@lists.sourceforge.net if you have questions regarding use of this software. To submit patches or suggest improvements, send email to nagiosplug-devel@lists.sourceforge.net

$ <b>cd /usr/local/nagios/etc</b> $ <b>ls -1</b>

cgi.cfg-sample

checkcommands.cfg-sample

contactgroups.cfg-sample

contacts.cfg-sample

dependencies.cfg-sample

escalations.cfg-sample

hostgroups.cfg-sample

hosts.cfg-sample

misccommands.cfg-sample

nagios.cfg-sample

resource.cfg-sample

```
services.cfg-sample

timeperiods.cfg-sample

# <b>for i in *cfg-sample; do mv $i `echo $i | \</b> <b>sed -e s/cfg-
sample/cfg/`; done;</b>

# <b>/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg</b>

hosts.cfg

services.cfg

contacts.cfg

contactgroups.cfg

hostgroups.cfg

# Generic host definition template define host{

# The name of this host template - referenced i name generic-host

n other host definitions, used for template recursion/resolution # Host
notifications are enabled notifications_enabled 1

# Host event handler is enabled event_handler_enabled 1

# Flap detection is enabled

flap_detection_enabled 1

# Process performance data

process_perf_data 1

# Retain status information across program restarts
retain_status_information 1
}

# Retain non-status information across program restarts
retain_nonstatus_information 1

# DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, #
JUST A TEMPLATE!

register 0

}

# Host Definition

define host{

# Name of host template to use use generic-host

host_name freelinuxcd.org

alias Free Linux CD Project Server address www.freelinuxcd.org

check_command check-host-alive max_check_attempts 10

notification_interval 120

notification_period 24x7

notification_options d,u,r

}

define hostgroup{

hostgroup_name flcd-servers

alias The Free Linux CD Project Servers contact_groups flcd-admins

members freelinuxcd.org

}
```

```
define contactgroup{

contactgroup_name flcd-admins alias FreeLinuxCD.org Admins members
oktay, verty

}

define contact{

contact_name oktay

alias Oktay Altunergil

service_notification_period 24x7

host_notification_period 24x7

service_notification_options w,u,c,r host_notification_options d,u,r
service_notification_commands notify-by-email,notify-by-epager
host_notification_commands host-notify-by-email,host-notify-by-epager
email oktay@freelinuxcd.org

pager dummypagenagios-admin@localhost.localdomain }

define contact{

contact_name Verty

alias David 'Verty' Ky

service_notification_period 24x7

host_notification_period 24x7

service_notification_options w,u,c,r host_notification_options d,u,r
service_notification_commands notify-by-email,notify-by-epager
host_notification_commands host-notify-by-email email verty@flcd.org

}
```

```
# Generic service definition template define service{

# The 'name' of this service template, referenced in other service definitions
name generic-service

# Active service checks are enabled active_checks_enabled 1

# Passive service checks are enabled/accepted passive_checks_enabled 1

# Active service checks should be parallelized # (disabling this can lead to
major performance problems) parallelize_check 1

# We should obsess over this service (if necessary) obsess_over_service 1

# Default is to NOT check service 'freshness'

check_freshness 0

# Service notifications are enabled notifications_enabled 1

# Service event handler is enabled event_handler_enabled 1

# Flap detection is enabled

flap_detection_enabled 1

# Process performance data

process_perf_data 1

# Retain status information across program restarts
retain_status_information 1

# Retain non-status information across program restarts
retain_nonstatus_information 1

# DONT REGISTER THIS DEFINITION - ITS NOT A REAL SERVICE,
JUST A TEMPLATE!
```

```
register 0

}

# Service definition

define service{

# Name of service template to use use generic-service

host_name freelinuxcd.org

service_description HTTP

is_volatile 0

check_period 24x7

max_check_attempts 3

normal_check_interval 5

retry_check_interval 1

contact_groups flcd-admins

notification_interval 120

notification_period 24x7

notification_options w,u,c,r check_command check_http

}

# Service definition

define service{

# Name of service template to use use generic-service
```

```
host_name freelinuxcd.org

service_description PING

is_volatile 0

check_period 24x7

max_check_attempts 3

normal_check_interval 5

retry_check_interval 1

contact_groups flcd-admins

notification_interval 120

notification_period 24x7

notification_options c,r

check_command check_ping!100.0,20%!500.0,60%

}

# <b>/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg</b>
```

That's all there is to it. Give Nagios a couple of minutes to generate some data, and then point your browser to the machine and look at the pretty service warning lights.

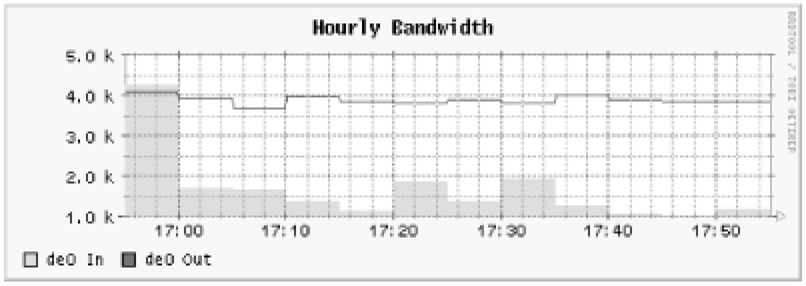
```
$ <b>rrdtool create zul.rrd --start N \</b>

<span class="docEmphBold">DS:de0_in:COUNTER:600:U:U \</span>
<span class="docEmphBold">DS:de0_out:COUNTER:600:U:U \</span>
<span class="docEmphBold">RRA:AVERAGE:0.5:1:600 \</span> <span
class="docEmphBold">RRA:AVERAGE:0.5:6:700 \</span> <span
class="docEmphBold">RRA:AVERAGE:0.5:24:775 \</span> <span
class="docEmphBold">RRA:AVERAGE:0.5:288:797 \</span> <span
class="docEmphBold">RRA:MAX:0.5:1:600 \</span> <span
class="docEmphBold">RRA:MAX:0.5:6:700 \</span> <span
class="docEmphBold">RRA:MAX:0.5:24:775 \</span> <span
class="docEmphBold">RRA:MAX:0.5:288:797</span>

$ <b>rrdtool update zul.rrd N:\</b>

<b>`snmpget -Oqv zul public interfaces.ifTable.ifEntry.ifInOctets.4`:\</b>
<b>`snmpget -Oqv zul public interfaces.ifTable.ifEntry.ifOutOctets.4`</b>
```

```
0-55/5 * * * * rrdtool update /home/andrew/rrdbs/zul.rrd N:`snmpget -Oqv
zul public interfaces.ifTable.ifEntry.ifInOctets.4`:`snmpget -Oqv zul public
interfaces.ifTable.ifEntry.ifOutOctets.4`
```

```
rrdtool graph zul_de0-hourly.png -t "Hourly Bandwidth" --start -3600 \
DEF:inoctets=zul.rrd:de0_in:AVERAGE \

   DEF:outoctets=zul.rrd:de0_out:AVERAGE \
AREA:inoctets#00FF00:"de0 In" \

   LINE1:outoctets#0000FF:"de0 Out"
```

This would create an image like the one shown in Figure 5-1.

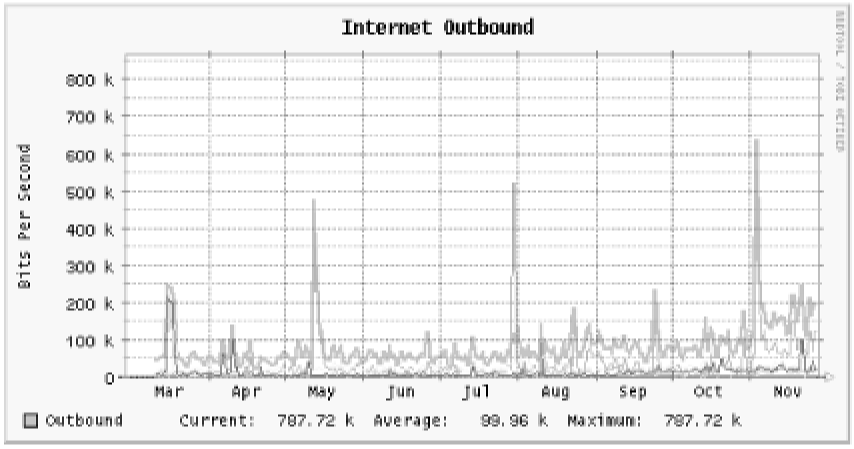
Figure 5-1. A graph generated by RRDtool



The `-3600` in the command tells `rrdtool` that you want to graph the data collected over the last hour (there are 3,600 seconds in an hour). Likewise, if you wanted to create a graph over the course of a day, you would use `-86400`.

But that's just the beginning. After collecting multiple data sources, you can combine them all into a single graph that gives you a great deal of information at a glance. Figure 5-2 shows the relative outbound usage of several servers simultaneously, with the total average for all servers just below it. While this figure is in grayscale, the actual graph uses a different color for each server, making it easy to tell at a glance which one is hogging all of the bandwidth.

Figure 5-2. Multiple servers on a single graph



As you can see, *RRDtool* is a very flexible tool. All you need to do is tell it how much data you want to store and then set up some method to collect the data at a regular interval. Then you can easily generate a graph of the data whenever you want it.

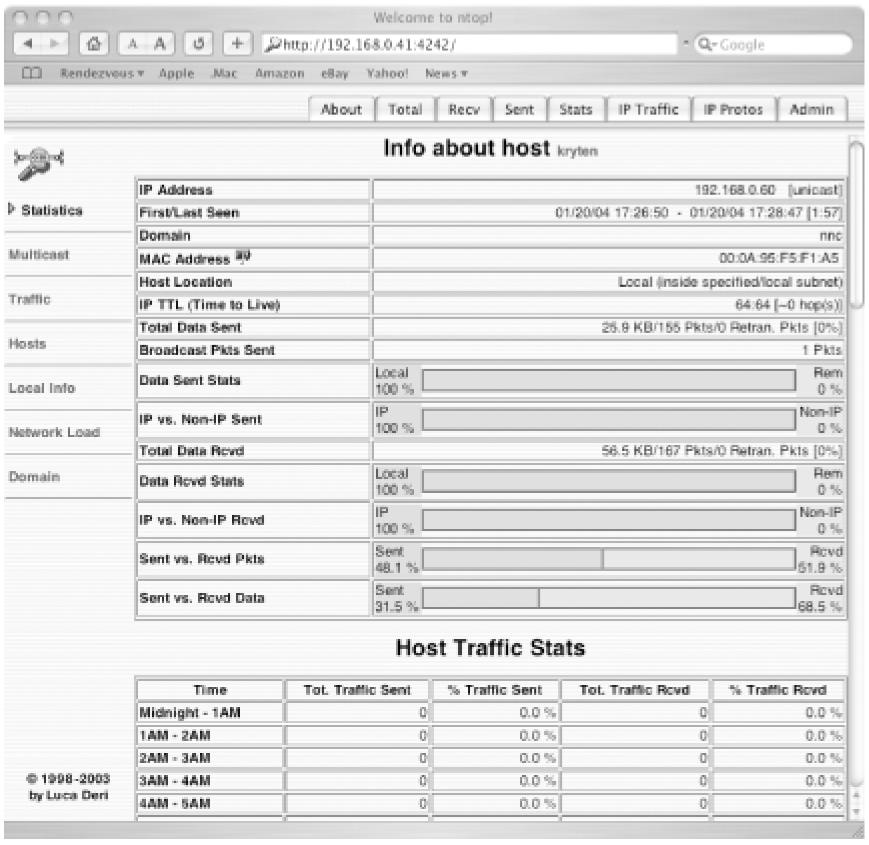
```
# <b>groupadd ntop</b>

# <b>useradd -c "ntop user" -d /usr/local/etc/ntop \</b>
    <b>-s /bin/true -g ntop ntop</b>
# <b>mkdir /usr/local/etc/ntop</b>
# <b>cp /usr/local/src/ntop-2.1.3/ntop/*pem /usr/local/etc/ntop</b>
# <b>ntop -A -u ntop -P /usr/local/etc/ntop</b>
21/Sep/2002 20:30:23 Initializing GDBM...

21/Sep/2002 20:30:23 Started thread (1026) for network packet analyser.

21/Sep/2002 20:30:23 Started thread (2051) for idle hosts detection.

21/Sep/2002 20:30:23 Started thread (3076) for DNS address resolution.

21/Sep/2002 20:30:23 Started thread (4101) for address purge.

Please enter the password for the admin user:

Please enter the password again:
```

```
21/Sep/2002 20:30:29 Admin user password has been set.

# <b>ntop -u ntop -P /usr/local/etc/ntop -W4242 -d</b>
```

By default, *ntop* also runs a standard HTTP server on port 3000. You should seriously consider locking down access to these ports, either at your firewall or by using command-line iptables rules [Hack #33] .

Let *ntop* run for a while, then connect to https://`your.server.here`:4242/. You can find out all sorts of details about what traffic has been seen on your network, as shown in Figure 5-3.

Figure 5-3. Displaying a host's statistics in ntop's web interface



Info about host kryten

IP Address	192.168.0.60 [unicast]
First/Last Seen	01/20/04 17:26:50 - 01/20/04 17:28:47 [1:57]
Domain	nnc
MAC Address	00:0A:95:F5:F1:A5
Host Location	Local (inside specified/local subnet)
IP TTL (Time to Live)	64:64 [~0 hop(s)]
Total Data Sent	25.9 KB/155 Pkts/0 Retran. Pkts [0%]
Broadcast Pkts Sent	1 Pkts
Data Sent Stats	Local 100 % / Rem 0 %
IP vs. Non-IP Sent	IP 100 % / Non-IP 0 %
Total Data Rcvd	56.5 KB/167 Pkts/0 Retran. Pkts [0%]
Data Rcvd Stats	Local 100 % / Rem 0 %
IP vs. Non-IP Rcvd	IP 100 % / Non-IP 0 %
Sent vs. Rcvd Pkts	Sent 48.1 % / Rcvd 51.9 %
Sent vs. Rcvd Data	Sent 31.5 % / Rcvd 68.5 %

Host Traffic Stats

Time	Tot. Traffic Sent	% Traffic Sent	Tot. Traffic Rcvd	% Traffic Rcvd
Midnight - 1AM	0	0.0 %	0	0.0 %
1AM - 2AM	0	0.0 %	0	0.0 %
2AM - 3AM	0	0.0 %	0	0.0 %
3AM - 4AM	0	0.0 %	0	0.0 %
4AM - 5AM	0	0.0 %	0	0.0 %

While tools like tcpdump and Ethereal give you detailed, interactive analysis of network traffic, *ntop* delivers a wealth of statistical information in a very slick and easy-to-use web interface. When properly installed and locked down, it will likely become a favorite tool in your network analysis tool chest

Rob Flickenger (Linux Server Hacks)

```
$ <b>tar xfz argus-2.0.5.tar.gz</b>

$ <b>cd argus-2.0.5</b>

$ <b>./configure && make</b>

# <b>make install</b>

# <b>argus -d -e `hostname` -w /tmp/arguslog</b>

$ <b>ra -r /tmp/arguslog</b>

12 Jan 04 05:42:48 udp plunder.nnc.netbios-ns -> 192.168.0.255.netbios-ns
INT

12 Jan 04 05:43:09 udp 192.168.0.250.snmptrap ->
255.255.255.255.snmptrap INT

12 Jan 04 05:43:15 udp print.nnc.netbios-dgm -> 192.168.0.255.netbios-
dgm INT

12 Jan 04 05:43:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:43:28 nvl 0:c0:2:57:98:79 -> Broadcast INT

12 Jan 04 05:43:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:43:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:44:19 udp kryten.nnc.56581 -> 255.255.255.255.2222 TIM

12 Jan 04 05:43:34 udp sunder.nnc.netbios-ns -> 192.168.0.255.netbios-ns
INT

12 Jan 04 05:44:08 arp plunder.nnc who-has sirius.nnc INT

12 Jan 04 05:44:08 udp plunder.nnc.netbios-ns -> 192.168.0.255.netbios-ns
INT
```

```
12 Jan 04 05:44:15 udp print.nnc.netbios-dgm -> 192.168.0.255.netbios-dgm INT

12 Jan 04 05:45:06 udp sunder.nnc.netbios-dgm -> 192.168.0.255.netbios-dgm TIM

12 Jan 04 05:40:26 man pkts 734 bytes 75574 drops 0 CON

12 Jan 04 05:44:28 nvl 0:c0:2:57:98:79 -> Broadcast INT

12 Jan 04 05:44:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:44:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:44:28 llc 0:c0:2:57:98:79.null -> Broadcast.null INT

12 Jan 04 05:45:08 udp plunder.nnc.netbios-ns -> 192.168.0.255.netbios-ns INT

12 Jan 04 05:45:09 tcp kryten.nnc.54176 ?> colossus.nnc.ssh EST

12 Jan 04 05:45:15 udp print.nnc.netbios-dgm -> 192.168.0.255.netbios-dgm INT

$ <b>ra -r /tmp/argus</b> <b>- "host kryten"</b> 12 Jan 04 09:26:34 udp kryten.nnc.55689 -> 255.255.255.255.2222 TIM

12 Jan 04 09:26:36 tcp kryten.nnc.54176 ?> linux-vmm.nnc.ssh EST

12 Jan 04 09:27:37 tcp kryten.nnc.54176 ?> linux-vmm.nnc.ssh EST

12 Jan 04 09:28:34 udp kryten.nnc.55691 -> 255.255.255.255.2222 TIM

12 Jan 04 09:28:05 icmp kryten.nnc <-> linux-vmm.nnc ECO

12 Jan 04 09:28:06 icmp kryten.nnc <-> linux-vmm.nnc ECO

12 Jan 04 09:29:06 tcp kryten.nnc.54176 ?> linux-vmm.nnc.ssh EST
```

```
12 Jan 04 09:30:34 udp kryten.nnc.55692 -> 255.255.255.255.2222 TIM

12 Jan 04 09:32:34 udp kryten.nnc.55693 -> 255.255.255.255.2222 TIM

12 Jan 04 09:33:06 tcp kryten.nnc.54176 ?> linux-vmm.nnc.ssh EST

12 Jan 04 09:34:34 udp kryten.nnc.55694 -> 255.255.255.255.2222

12 Jan 04 09:53:44 tcp kryten.nnc.54176 ?> linux-vmm.nnc.ssh EST

$ <b>raxml -r /tmp/arguslog - "host kryten"</b> <ArgusFlowRecord
ArgusSourceId = "192.168.0.41" SequenceNumber = "3"

Cause = "Status" StartDate = "2004-01-12" StartTime = "09:25:26"

StartTimeusecs = "319091" LastDate = "2004-01-12"

LastTime = "09:25:32" LastTimeusecs = "521982"

Duration = "6.202891" TransRefNum = "0">

   <MACAddrs SrcAddr = "0:a:95:c7:2b:10" DstAddr = "0:c:29:e2:2b:c1"
/> <Flow> <IP SrcIPAddr = "192.168.0.60" DstIPAddr = "192.168.0.41"

   Proto = "tcp" Sport = "56060" Dport = "22" IpId = "27b8" /> </Flow>
<FlowAttrs SrcTTL = "64" DstTTL = "64" SrcTOS = "10" DstTOS = "10"
/> <ExtFlow> <TCPExtFlow TCPState = "EST" TCPOptions = "TIME"

   SynAckuSecs = "0" AckDatauSecs = "0" > <TCPExtMetrics
SrcTCPSeqBase = "4204580547"

   SrcTCPAckBytes = "527" SrcTCPBytes = "528"

   SrcTCPRetrans = "0" SrcTCPWin = "65535" SrcTCPFlags = "PA"

   DstTCPSeqBase = "3077608383" DstTCPAckBytes = "1135"

   DstTCPBytes = "992" DstTCPRetrans = "0" DstTCPWin = "9792"
```

```
    DstTCPFlags = "PA" />

    </TCPExtFlow>

    </ExtFlow>

    <Metrics SrcCount = "24" DstCount = "17" SrcBytes = "2112"

    DstBytes = "2258" SrcAppBytes = "528" DstAppBytes = "1136" />
</ArgusFlowRecord>
```

As you can see, Argus keeps track of much more information than it would seem if you were just going by the output generated by `ra`. This is where Argus really shines, because it can store such a large amount of information about your network traffic in a small amount of space. In addition, Argus makes it easy to convert this information into other formats, such as XML, which makes it easy to write applications that can understand the data.

```
# <b>iptables -N KRYTEN && iptables -A KRYTEN -j ACCEPT</b>

# <b>iptables -N KRYTEN_IN && iptables -A KRYTEN_IN -j
KRYTEN</b>

# <b>iptables -N KRYTEN_OUT && iptables -A KRYTEN_OUT -j
KRYTEN</b>

# <b>iptables -A FORWARD -s 192.168.0.60</b> <b>-j
KRYTEN_OUT</b>

# <b>iptables -A FORWARD -d 192.168.0.60 -j KRYTEN_IN</b>

# <b>iptables -vx -L KRYTEN</b>



Chain kryten (2 references)



pkts bytes target prot opt in out source destination



   442 46340 ACCEPT all -- any any anywhere anywhere

# <b>iptables -vx -L KRYTEN | egrep -v 'Chain|pkts' | awk '{print $2}'</b>
```

To get the inbound or outbound bandwidth consumed, just replace `KRYTEN` with `KRYTEN_IN` or `KRYTEN_OUT`, respectively. Of course, you don't have to limit your statistic collection criteria to just per-computer bandwidth usage. You can collect statistics on anything that you can create an iptables rule for, including ports, MAC addresses, or just about anything else that passes through your network.

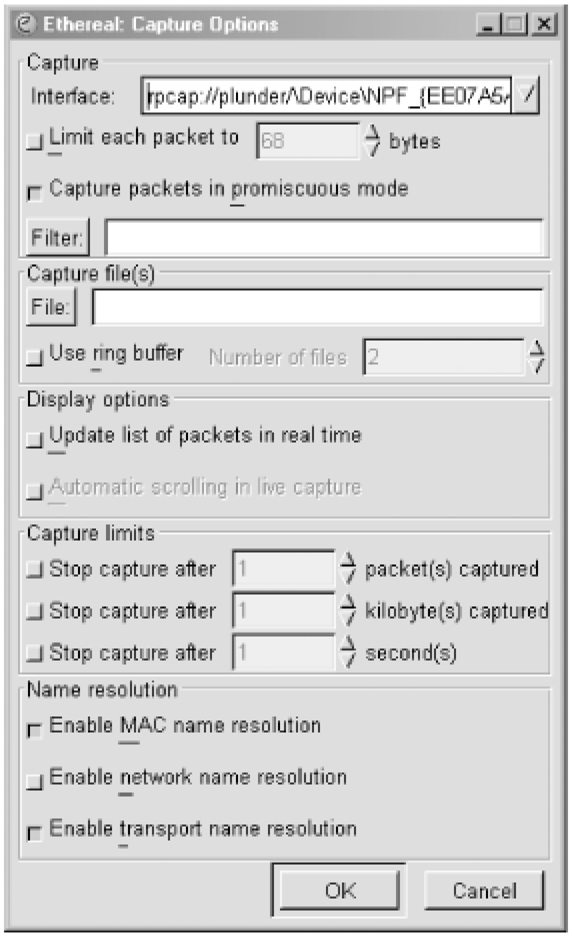
```
C:\Program Files\WinPcap><b>rpcapd -l obsidian -n</b>
Press CTRL + C to stop the server...
C:\Program Files\WinPcap><b>rpcapd -l obsidian -n -s rpcapd.ini</b>
Press CTRL + C to stop the server...
C:\Program Files\WinPcap><b>type rpcapd.ini</b>
# Configuration file help.

# Hosts which are allowed to connect to this server (passive mode)
# Format: PassiveClient = <name or address>
PassiveClient = obsidian

# Hosts to which this server is trying to connect to (active mode)
# Format: ActiveClient = <name or address>, <port | DEFAULT>
# Permit NULL authentication: YES or NOT

NullAuthPermit = YES
C:\Program Files\WinPcap><b>net start rpcapd</b>
The Remote Packet Capture Protocol v.0 (experimental) service was started
successfully.
C:\Program Files\WinPcap><b>windump -D</b>
```

1.\Device\NPF_{EE07A5AE-4D19-4118-97CE-3BF656CD718F} (NDIS 5.0 driver)

rpcap://plunder/\Device\NPF_{EE07A5AE-4D19-4118-97CE-3BF656CD718F}

You can see an example of this with Ethereal in Figure 5-5.

Figure 5-5. Using a remote capture source with Ethereal


Ethereal: Capture Options
Capture
Interface: rpcap://plunder/\Device\NPF_{EE07A5
Limit each packet to 68 bytes
Capture packets in promiscuous mode
Filter:
Capture file(s)
File:
Use ring buffer
Number of files 2
Display options
Update list of packets in real time
Automatic scrolling in live capture
Capture limits
Stop capture after 1 packet(s) captured
Stop capture after 1 kilobyte(s) captured
Stop capture after 1 second(s)
Name resolution
Enable MAC name resolution
Enable network name resolution
Enable transport name resolution
OK
Cancel

If you've set up everything correctly, you should see traffic streaming from the remote end into your sniffer just as if it were being captured from a local interface.

Chapter 6. Secure Tunnels

Hacks #67-81

Untrusted computer networks (such as the Internet and public wireless networks) can be pretty hostile environments, but they can be tamed to some degree. By

leveraging encryption and some encapsulation tricks, you can build more trustworthy networks on top of whatever network you choose, even if it is full of miscreants trying to watch or otherwise manipulate your data. This chapter primarily

deals with how to set up secure, encrypted communications over networks that you don't completely trust. Some of the hacks focus mainly on providing a secure and encrypted transport mechanism, while others discuss how to create a virtual private network (VPN).

In reading this chapter, you'll learn how to set up Ipsec-based encrypted links on several operating systems, how to create virtual network interfaces that can be tunneled through an encrypted connection, and how to forward TCP connections over an encrypted channel. In addition,

you'll also learn how to set up a cross-platform VPN

solution.

The beauty of most of these hacks is that after reading them, you can mix and match transport-layer encryption solutions with whatever virtual network-oriented approach that best meets your needs. In this way, you can safely

build vast, powerful private networks leveraging the public Internet as infrastructure. You can use these

techniques for anything from securely connecting two remote offices to building a completely routed private network enterprise on top of the Internet.

```
# /etc/ipsec.conf

# Set configuration options
```

```
config setup

   interfaces=%defaultroute # Debug parameters. Set either to "all" for more
info klipsdebug=none plutodebug=none # standard Pluto configuration
plutoload=%search plutostart=%search # make sure there are no PMTU
Discovery problems overridemtu=1443

# default configuration settings conn %default

   # Be aggressive in rekeying attempts keyingtries=0

   # use IKE

   keyexchange=ike keylife=12h # use shared secrets authby=secret # setup
the VPN to the Internet

conn wireless_connection1

   type=tunnel # left is the client side left=192.168.0.104

   # right is the internet gateway right=192.168.0.1

   rightsubnet=0.0.0.0/0

   # automatically start the connection auto=start

192.168.0.104 192.168.0.1: PSK "supersecret"

# assume internal ethernet interface is eth0

interfaces="ipsec0=eth0"

...

conn wireless_connection2

   type=tunnel left=192.168.0.105
   right=192.168.0.1

   rightsubnet=0.0.0.0/0

   auto=start conn wireless_connection3

   type=tunnel left=192.168.0.106

   right=192.168.0.1

   rightsubnet=0.0.0.0/0

   auto=start ...
```

Finally, add the shared secrets for all the clients to ipsec.secrets:
192.168.0.105 192.168.0.1: PSK "evenmoresecret"

192.168.0.106 192.168.0.1: PSK "notsosecret"

Clients should now be connecting to the Internet via a VPN tunnel to the gateway. Check the log files or turn up the debug level if the tunnel does not come up.

```
options IPSEC #IP security

options IPSEC_ESP #IP security (crypto; define w/ IPSEC) options
IPSEC_DEBUG #debug for IP security

path include "/usr/local/etc/racoon" ; path pre_shared_key
"/usr/local/etc/racoon/psk.txt" ; remote anonymous

{

   exchange_mode aggressive,main; my_identifier user_fqdn
"user1@domain.com"; lifetime time 1 hour; initial_contact on; proposal {

   encryption_algorithm 3des; hash_algorithm sha1; authentication_method
pre_shared_key ; dh_group 2 ; }

}
```

```
sainfo anonymous

{

   pfs_group 1; lifetime time 30 min; encryption_algorithm 3des ;
authentication_algorithm hmac_sha1; compression_algorithm deflate ; }

#!/bin/sh

# This script will start racoon in FreeBSD

case "$1" in

start)

# start racoon

   echo -n 'starting racoon'

   /usr/local/sbin/racoon ;;

stop)

# Delete the MAC address from the ARP table echo 'stopping racoon'

   killall racoon ;;

*)

# Standard usage statement

   echo "Usage: `basename $0` {start|stop}" >&2

   ;;
```

esac

exit 0

# <b>chmod 755 /usr/local/etc/rc.d/racoon.sh</b>

user1@domain.com supersecret

# <b>spdadd 192.168.0.104/32 0.0.0.0/0 any -P out ipsec \ </b> <b>esp/tunnel/192.168.0.104-192.168.0.1/require ; </b> # <b>spdadd 0.0.0.0/0 192.168.0.104/32 any -P in ipsec \ </b> <b>esp/tunnel/192.168.0.1-192.168.0.104/require ;</b>

# <b>setkey -f client.spd</b>

user1@domain.com supersecret

user2@domain.com evenmoresecret

user3@domain.com notsosecret

# <b>spdadd 0.0.0.0/0 192.168.0.104/32 any -P out ipsec \ </b> <b>esp/tunnel/192.168.0.1-192.168.0.104/require ; </b> # <b>spdadd 192.168.0.104/32 0.0.0.0/0 any -P in ipsec \ </b> <b>esp/tunnel/192.168.0.104-192.168.0.1/require ; </b> # <b>spdadd 0.0.0.0/0 192.168.0.105/32 any -P in ipsec \ </b> <b>esp/tunnel/192.168.0.1-192.168.0.105/require ; </b> # <b>spdadd 192.168.0.105/32 0.0.0.0/0 any -P out \</b> <b>ipsec esp/tunnel/192.168.0.105-192.168.0.1/require ; </b> # <b>spdadd 0.0.0.0/0 192.168.0.106/32 any -P in ipsec \ </b> <b>esp/tunnel/192.168.0.1-192.168.0.106/require ; </b> # <b>spdadd 192.168.0.106/32 0.0.0.0/0 any -P out ipsec \ </b> <b>esp/tunnel/192.168.0.106-192.168.0.1/require ; </b>

Load the SPD by issuing `setkey -f gateway.spd`. Verify the SPD entries using the `spddump` command in `setkey`. At this point, you should be able to ping a client from the gateway. It may take a packet or two for the VPN negotiation to complete, but the connection should be solid after that. If you are unable to ping, examine your syslog output for errors and warnings.

```
KeyNote-Version: 2

Authorizer: "POLICY"

Licensees: "passphrase:mypassword"

Conditions: app_domain == "IPsec policy" && esp_present == "yes" &&
esp_enc_alg == "aes" && esp_auth_alg == "hmac-sha" -> "true";

[General]

Listen-on= 192.168.1.1

Shared-SADB= Defined [Phase 1]

Default= ISAKMP-peer-remote #Default= ISAKMP-peer-remote-
aggressive [Phase 2]

Passive-Connections=IPsec-local-remote [ISAKMP-peer-remote]

Phase= 1

Transport= udp

Local-address= 192.168.1.1

Configuration= Default-main-mode Authentication= mypassword
[ISAKMP-peer-remote-aggressive]

Phase= 1

Transport= udp

Local-address= 192.168.1.1

Configuration= Default-aggressive-mode Authentication= mypassword
[IPsec-local-remote]
```

```
Phase= 2

ISAKMP-peer= ISAKMP-peer-remote Configuration= Default-quick-mode
Local-ID= Net-local Remote-ID= Net-remote [Net-remote]

ID-type= IPV4_ADDR

Address= 0.0.0.0

[Net-local]

ID-type= IPV4_ADDR

Address= 0.0.0.0

[Default-main-mode]

DOI= IPSEC

EXCHANGE_TYPE= ID_PROT

Transforms= 3DES-SHA [Default-aggressive-mode]

DOI= IPSEC

EXCHANGE_TYPE= AGGRESSIVE

Transforms= 3DES-SHA-RSA [Default-quick-mode]

DOI= IPSEC

EXCHANGE_TYPE= QUICK_MODE

Suites= QM-ESP-AES-SHA-PFS-SUITE

# <b>/sbin/isakmpd</b>

isakmpd_flags=""
```

That should do it. As usual, check your system logs if your tunnel has trouble connecting.

```
$ <b>./configure && make</b>

# <b>make install</b>

option /etc/ppp/options.pptpd


localip 10.0.0.1


remoteip 10.0.0.2-100
```

lock

name pptpd

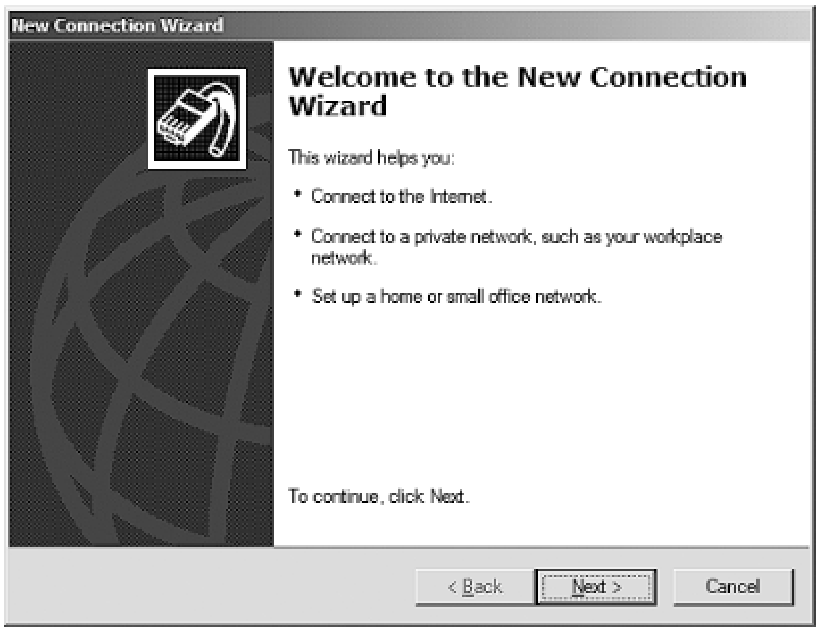
```
auth

# Secrets for authentication using CHAP


# client server secret IP addresses
```

```
andrew pptpd mypassword *
```

The `pptpd` in the server field should be replaced with whatever you used in the `name` directive in your */etc/ppp/options.pptpd* file (if you didn't use `pptpd`). You can of course limit the client to specific IP addresses by listing them.

Now that you have a basic setup for *PoPToP* , you can try it out by connecting to it with a Windows machine. Go to your Network Connections folder and click "Create a new connection" (this is for Windows XP; for Windows 2000, look for "Make New Connection"). After you click this, a wizard dialog should appear that looks similar to Figure 6-1.

Figure 6-1. Windows XP's New Connection Wizard


New Connection Wizard

Welcome to the New Connection Wizard

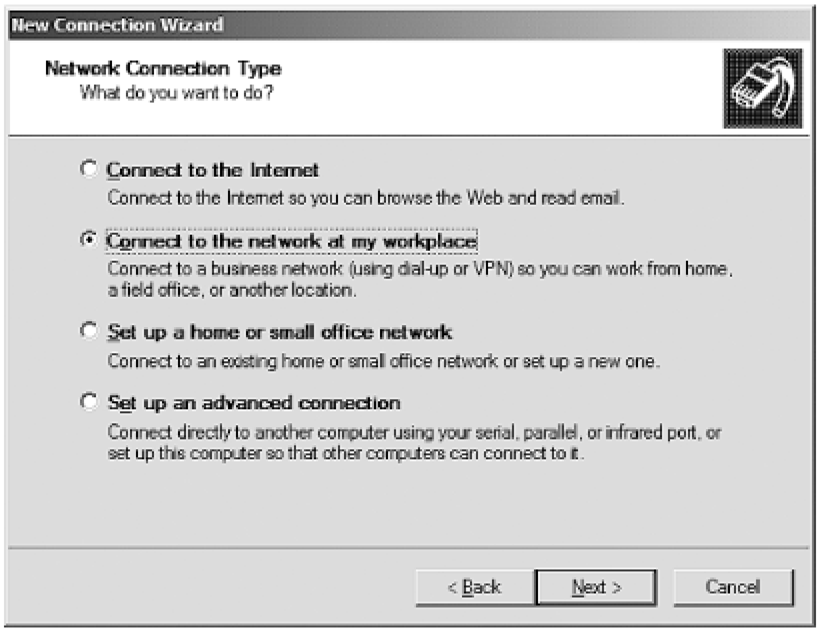
This wizard helps you:

- Connect to the Internet.
- Connect to a private network, such as your workplace network.
- Set up a home or small office network.

To continue, click Next.

Click Next and then select the "Connect to the network at my workplace" radio button, as shown in Figure 6-2.

Figure 6-2. Choosing the connection type


New Connection Wizard

Network Connection Type
What do you want to do?

( ) Connect to the Internet
Connect to the Internet so you can browse the Web and read email.

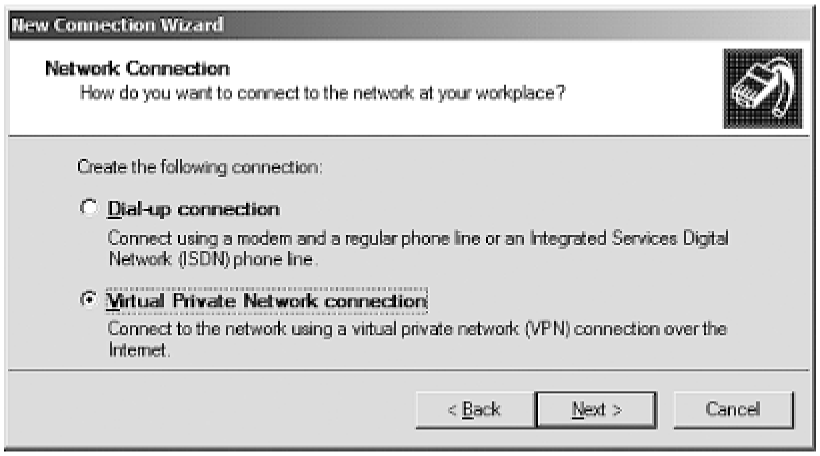
(•) Connect to the network at my workplace
Connect to a business network (using dial-up or VPN) so you can work from home, a field office, or another location.

( ) Set up a home or small office network
Connect to an existing home or small office network or set up a new one.

( ) Set up an advanced connection
Connect directly to another computer using your serial, parallel, or infrared port, or set up this computer so that other computers can connect to it.

After you've done that, click Next again and then click the "Virtual Private Network connection" radio button. You should now see something similar to Figure 6-3.

Figure 6-3. Selecting a VPN connection


New Connection Wizard

Network Connection
How do you want to connect to the network at your workplace?

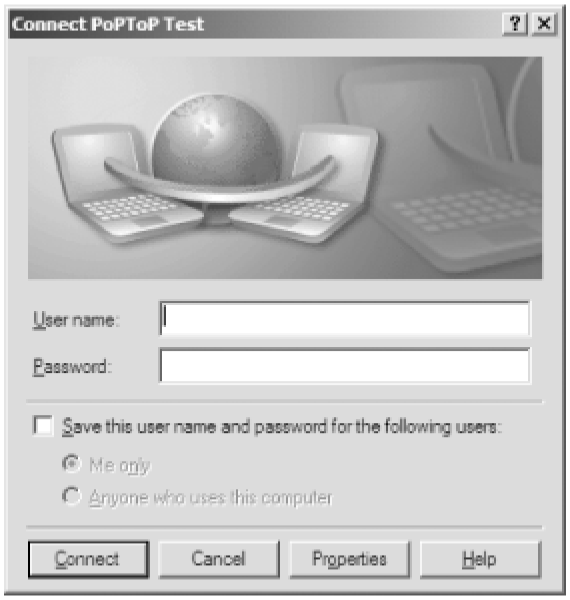
Create the following connection:

( ) Dial-up connection
Connect using a modem and a regular phone line or an Integrated Services Digital Network (ISDN) phone line.

(•) Virtual Private Network connection
Connect to the network using a virtual private network (VPN) connection over the Internet.

Click Next and fill in a name for the newly created connection (e.g., PoPToP Test). After you've done that, click Next once again and then enter the external IP address of the server running *pptpd*. Now click Next and then Finish. You'll then be presented with a login dialog similar to the one shown in Figure 6-4.

Figure 6-4. The connection login dialog


Connect PoPToP Test

User name:

Password:

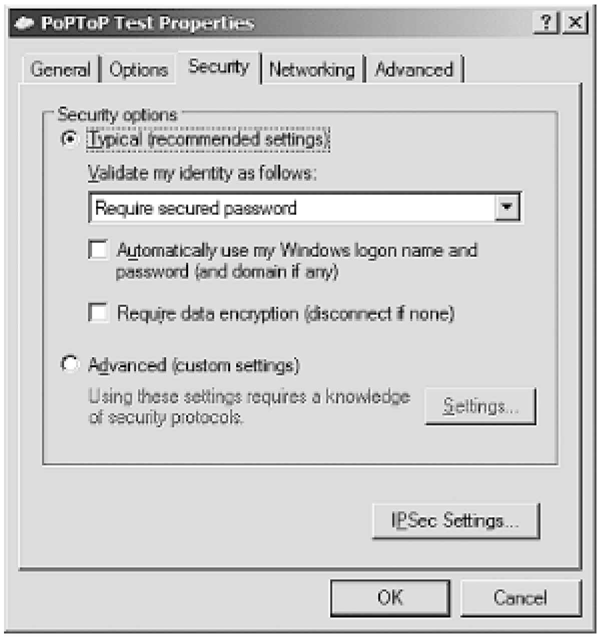
[ ] Save this user name and password for the following users:

(●) Me only

( ) Anyone who uses this computer

Before entering the username and password that you specified in the */etc/ppp/chap-secrets* file, you'll need to click Properties and locate the Security tab. After you've done that, locate the "Require data encryption" checkbox and uncheck it. You should now see something similar to Figure 6-5.

Figure 6-5. Changing the security properties


PoPToP Test Properties

Security options

(●) Typical (recommended settings)

Validate my identity as follows:

Require secured password

[ ] Automatically use my Windows logon name and password (and domain if any)

[ ] Require data encryption (disconnect if none)

( ) Advanced (custom settings)

Using these settings requires a knowledge of security protocols.

Now click OK, enter your login information, and then click Connect. In a few seconds you should be connected to the PPTP server and will be allocated an IP address from the pool that you specified. You should now test the connection by pinging the remote end of the tunnel. With the PPTP connection active, all traffic leaving the client side will be encrypted and sent to the *PoPToP* server. From there, traffic will make its way to its ultimate destination.

```
# <b>ipsec newhostkey --output /tmp/`hostname`.key</b>

# <b>cat /tmp/`hostname`.key >> /etc/ipsec.secrets</b>

# <b>ipsec showhostkey --txt @colossus.nnc</b> ; RSA 2192 bits colossus
Mon Jan 12 03:02:07 2004

   IN TXT "X-IPsec-Server(10)=@colossus.nnc" "

AQOR7rM7ZMBXu2ej/1vtzhNnMayZO1jwVHUyAIubTKpd/

PyTMogJBAdbb3I0xzGLaxadPGfiqPN2AQn76zLIsYFMJnoMbBTDY/2x
K1X/

pWFRUUIHzJUqCBIijVWEMLNrIhdZbei1s5/

MgYIPaX20UL+yAdxV4RUU3JJQhV7adVzQqEmdaNUnCjZOvZG6m4z
v6dGROrVEZmJFP54v6WhckYf
qSkQu3zkctfFgzJ/rMTB6Y38yObyBg2HuWZMtWI"

"8VrTQqi7IGGHK+mWk+wSoXer3iFD7JxRTzPOxLk6ihAJMibtKna3j7Q
P9ZHG0nm7NZ/

L5M9VpK+Rfe+evUUMUTfAtSdlpus2BIeXGWcPfz6rw305H9"

# <b>ipsec verify</b>

Checking your system to see if IPsec got installed and started correctly
Version check and ipsec on-path [OK]

Checking for KLIPS support in kernel [OK]

Checking for RSA private key (/etc/ipsec.secrets) [OK]

Checking that pluto is running [OK]

DNS checks.
```

```
Looking for TXT in forward map: colossus [OK]

Does the machine have at least one non-private address [OK]

# <b>/etc/init.d/ipsec restart</b>

# <b>ipsec showhostkey --txt 192.168.0.64</b> ; RSA 2192 bits colossus
Tue Jan 13 03:02:07 2004

    IN TXT "X-IPsec-Server(10)=192.168.0.64" "

AQOR7rM7ZMBXu2ej/1vtzhNnMayZO1jwVHUyAIubTKpd/

PyTMogJBAdbb3I0xzGLaxadPGfiqPN2AQn76zLIsYFMJnoMbBTDY/2x
K1X/

pWFRUUIHzJUqCBIijVWEMLNrIhdZbei1s5/

MgYIPaX20UL+yAdxV4RUU3JJQhV7adVzQqEmdaNUnCjZOvZG6m4z
v6dGROrVEZmJFP54v6WhckYf
qSkQu3zkctfFgzJ/rMTB6Y38yObyBg2HuWZMtWI"

"8VrTQqi7IGGHK+mWk+wSoXer3iFD7JxRTzPOxLk6ihAJMibtKna3j7Q
P9ZHG0nm7NZ/

L5M9VpK+Rfe+evUUMUTfAtSdlpus2BIeXGWcPfz6rw305H9"
```

Add this record to the reverse zone for your subnet, and other machines will be able to initiate opportunistic encryption with your machine. With opportunistic encryption in use, all traffic between the hosts will be automatically encrypted, protecting all services simultaneously. Pretty neat, huh?

```
# <b>ssh -f -N -L 110:mailhost:110 </b>

<span class="docEmphBoldItalic">user</span>

<b>@</b>


<span class="docEmphBoldItalic">mailhost</span>

# <b>ssh -f -N -L 110:mailhost:110 -L 25:mailhost:25 </b>

<span class="docEmphBoldItalic">user</span>

<b>@</b>


<span class="docEmphBoldItalic">mailhost</span>

rob@catlin:~$



rob@catlin:~$ <b>~C </b> (<i> it doesn't echo)</i>

ssh> <b>-L8080:localhost:80 </b>

Forwarding port.

$ <b>ssh -f -g -N -L8000:localhost:80 10.42.4.6</b>

$ <b>ssh -f -N -L5150:intranet.insider.nocat:80 gateway.nocat.net</b>
```

Assuming that you're running a private domain called *.nocat*, and that *gateway.nocat.net* also has a connection to the private network, all traffic to port 5150 of remote will be obligingly forwarded to *intranet.insider.nocat:80*. The address *intranet.insider.nocat* doesn't have to resolve in DNS to remote; it isn't looked up until the connection is made to *gateway.nocat.net*, and then it's gateway that does the lookup. To securely browse that site from remote, try connecting to *http://localhost:5150/* .

Rob Flickenger (Linux Server Hacks)

```
$ <b>ssh-keygen -t rsa</b>

Generating public/private rsa key pair.


Enter file in which to save the key (/home/rob/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:


Your identification has been saved in /home/rob/.ssh/id_rsa.


Your public key has been saved in /home/rob/.ssh/id_rsa.pub.


The key fingerprint is:


a6:5c:c3:eb:18:94:0b:06:a1:a6:29:58:fa:80:0a:bc rob@localhost

$ <b>cat .ssh/id_rsa.pub |</b> \

<b>ssh</b>


<span class="docEmphBoldItalic">server</span>

<b>"mkdir .ssh && chmod 0700 .ssh && cat > .ssh/authorized_keys2"</b>
```

Of course, substitute your server name for `server`. Now, simply `ssh server` and it should log you in automatically, without a password. And yes, it will use your shiny new public key for `scp`, too.

If this didn't work for you, check your file permissions on both *~/.ssh/** and *server:~/.ssh/**. Your private key (*id_rsa*) should be mode 0600 (and be present only on your local machine), and everything else should be mode 0655 or better. In addition, your home directory on the server will need to be mode 755 or better. If it is group writable, someone that belongs to the group that owns your home directory could remove *~/.ssh*, even if *~/.ssh* is not writable by that group. This might not seem obvious at first, but if they can do that, then they can create their own *~/.ssh* and an *authorized_keys2* file, which could contain whatever keys they wish. Luckily, the SSH daemon will catch this and deny public key authentication until your permissions are fixed.

Security Concerns

Some consider the use of public keys a potential security risk. After all, one only has to steal a copy of your private key to obtain access to your servers. While this is true, the same is certainly true of passwords.

Ask yourself, how many times a day do you enter a password to gain shell access to a machine (or `scp` a file)? How frequently is it the same password on many (or all) of those machines? Have you ever used that password in a way that might be questionable (on a web site, a personal machine that isn't quite up-to-date, or possibly with an SSH client on a machine that you don't directly control)? If any of these possibilities sound familiar, then consider that an SSH key in the same setting would make it virtually impossible for an attacker to later gain unauthorized access (providing, of course, that you keep your private key safe).

Another way to balance ease of use with security is to use a passphrase on your key, but use the SSH agent to manage your keys for you. When you start the agent, it will ask you for your passphrase once, and will cache it until you kill the agent. Some people even go as far as to store their SSH keys on removable media (such as a USB key chain), and take their keys with them wherever they go. However you choose to use SSH keys, you'll almost certainly find that they're a very useful alternative to traditional passwords.

Rob Flickenger (Linux Server Hacks)

```
rob@caligula:~$ <b>ssh -L 3128:localhost:3128 proxy.example.com -f -N</b>
```

This will set up an SSH tunnel and fork into the background automatically. Next, change the HTTP Proxy host in your browser to localhost, and reload your page. As long as your SSH tunnel is running, your web traffic will be encrypted all the way to *proxy.example.com*, where it is decrypted and sent on to the Internet.

The biggest advantage of this technique (compared to using the SSH SOCKS 4 proxy [Hack #75]) is that virtually all browsers support the use of HTTP proxies, while not every browser supports SOCKS 4. Also, if you are using Mac OS X, there is support for HTTP proxies built into the OS itself. This means that every properly written application will use your proxy settings transparently.

Note that HTTP proxies have the same difficulties with DNS as a SOCKS 4 proxy, so keep those points in mind when using your proxy. Typically, your *squid* proxy is used from a local network, so you don't usually run into the DNS schizophrenia issue. But your *squid* can theoretically run anywhere (even behind a remote firewall), so be sure to check out the notes on DNS in [Hack #75] .

Running *squid* takes a little bit of preparation, but it can both secure and accelerate your web traffic when using wireless. Of course, *squid* will support as many simultaneous wireless users as you care to throw at it, so be sure to set it up for all of your regular wireless users, and keep your web traffic private

Rob Flickenger (Wireless Hacks)

```
rob@caligula:~$ <b>ssh -D 8080 </b>

<span class="docEmphBoldItalic">remote</span>

192.168.1.10 intranet.example.com
```

Likewise, you can list any number of hosts that are reachable only from the inside of your corporate firewall. When you attempt to browse to one of those sites, the local *hosts* file is consulted before DNS, so the private IP address is used. Since this request is actually made from *`remote`*, it finds its way to the internal server with no trouble. Likewise, responses arrive back at the SOCKS proxy on *`remote`*, are encrypted and forwarded over your SSH tunnel, and appear in your browser as if they came in from the Internet.

SOCKS 5 support is planned for an upcoming version of SSH, which will also make tunneled DNS resolution possible. This is particularly exciting for Mac OS X users, as there is support in the OS for SOCKS 5 proxies. Once SSH supports SOCKS 5, every native OS X application will automatically be able to take advantage of encrypting SSH socks proxies. In the meantime, we'll just have to settle for encrypted HTTP proxies [Hack #74] .

Rob Flickenger (Wireless Hacks)

```
$ <b>./configure --with-tcp-wrappers --with-ssl=/opt/openssl</b>

pid =


client = yes


[<server port>]


accept = <forwarded port>

connect = <remote address>:<server port>

cert = /etc/stunnel/stunnel.pem

pid =


client = no


[<forwarded port>]


accept = <server port>

connect = <forwarded port>

ssh -f -N -L <<tt><i>forwarded port</i></tt>>:<<tt><i>remote address</i>
</tt>>:<<tt><i>forwarded port</i></tt>> \
```

<<tt><i>remote address</i></tt>>

```
swat stream tcp nowait.400 root /usr/local/samba/bin/swat swat

cert = /etc/stunnel/swat.pem

exec = /usr/local/samba/bin/swat

execargs = swat

swat stream tcp nowait.400 root /usr/sbin/stunnel stunnel \

/etc/stunnel/swat.conf

cert = /etc/stunnel/swat.pem

[swat]


accept = 901


exec = /usr/local/samba/bin/swat

execargs = swat

# stunnel /etc/stunnel/swat.conf
```

In addition, you can start it at boot time by putting the previous command in your startup scripts (i.e., */etc/rc.local*).

Stunnel is a very powerful tool: not only can it forward connections through an encrypted tunnel, but it can also be used to add SSL capabilities to common services. This is especially nice when clients with SSL support for these services already exist. Thus, you can use *stunnel* solely on the server side, enabling encryption for the service with no need for the client to install any extra software.

```
$ <b>tar xfz httptunnel-3.3.tar.gz</b>

$ <b>cd httptunnel-3.3</b>

$ <b>./configure && make</b>

# <b>hts -F localhost:22 80</b>

# <b>htc -F 2222 colossus:80</b>

# <b>/usr/sbin/lsof -i | grep htc</b>

htc 2323 root 6u IPv4 0x02358a30 0t0 TCP *:2222 (LISTEN)

[andrew@kryten andrew]$ <b>ssh -p 2222 localhost</b>

andrew@localhost's password:


[andrew@colossus andrew]$

# <b>hts -F oceana.ingsoc.net:22 80</b>

POST /index.html?crap=1071364879 HTTP/1.1


Host: linux-vm:80


Content-Length: 102400


Connection: close
```

```
SSH-2.0-OpenSSH_3.6.1p1+CAN-2003-0693
```

```
<span class="docEmphBold">htc -P myproxy:8000 -A
andrew:mypassword -F 22 colossus:80</span>
```

If the port that the proxy listens on is the standard web proxy port (8080), then you can just specify the proxy by using its IP address or hostname.

```
root@client:~# <b>ifconfig eth2</b> eth2 Link encap:Ethernet HWaddr
00:02:2D:2A:27:EA inet addr:10.42.3.2 Bcast:10.42.3.63
Mask:255.255.255.192

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:662 errors:0 dropped:0 overruns:0 frame:0

TX packets:733 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:100

RX bytes:105616 (103.1 Kb) TX bytes:74259 (72.5 Kb) Interrupt:3 Base
address:0x100

root@client:~# route
```

```
Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface 10.42.3.0 *
255.255.255.192 U 0 0 0 eth2

loopback * 255.0.0.0 U 0 0 0 lo

default 10.42.3.1 0.0.0.0 UG 0 0 0 eth2

root@client:~# <b>traceroute -n yahoo.com</b> traceroute to yahoo.com
(64.58.79.230), 30 hops max, 40 byte packets 1 10.42.3.1 2.848 ms 2.304
ms 2.915 ms

2 209.204.179.1 16.654 ms 16.052 ms 19.224 ms 3 208.201.224.194 20.112
ms 20.863 ms 18.238 ms 4 208.201.224.5 213.466 ms 338.259 ms 357.7 ms
5 206.24.221.217 20.743 ms 23.504 ms 24.192 ms 6 206.24.210.62 22.379
ms 30.948 ms 54.475 ms 7 206.24.226.104 94.263 ms 94.192 ms 91.825 ms
8 206.24.238.61 97.107 ms 91.005 ms 91.133 ms 9 206.24.238.26 95.443
ms 98.846 ms 100.055 ms 10 216.109.66.7 92.133 ms 97.419 ms 94.22 ms
11 216.33.98.19 99.491 ms 94.661 ms 100.002 ms 12 216.35.210.126
97.945 ms 93.608 ms 95.347 ms 13 64.58.77.41 98.607 ms 99.588 ms
97.816 ms

# <b>modprobe tun</b>

options {

   port 5000;

   ifconfig /sbin/ifconfig; route /sbin/route; syslog auth;

}

default {

   compress no;

   speed 0;
```

```
}

home {

   type tun;

   proto tcp;

   stat yes;

   keepalive yes;

   pass sHHH; # Password is REQUIRED.

   up {

   ifconfig "%% 208.201.239.32 pointopoint 208.201.239.33"; program
/sbin/arp "-Ds 208.201.239.33 %% pub"; program /sbin/arp "-Ds
208.201.239.33 eth0 pub"; route "add -net 10.42.0.0/16 gw
208.201.239.33"; };

   down {

   program /sbin/arp "-d 208.201.239.33 -i %%"; program /sbin/arp "-d
208.201.239.33 -i eth0"; route "del -net 10.42.0.0/16 gw 208.201.239.33";
};

}

root@server:~# <b>vtund -s</b>

options {

   port 5000;

   ifconfig /sbin/ifconfig; route /sbin/route; }

default {
   compress no;

   speed 0;

}

home {

   type tun;

   proto tcp;

   keepalive yes;

   pass sHHH; # Password is REQUIRED.

   up {

   ifconfig "%% 208.201.239.33 pointopoint 208.201.239.32 arp"; route
"add 208.201.239.5 gw 10.42.3.1"; route "del default"; route "add default
gw 208.201.239.32"; };

   down {

   route "del default"; route "del 208.201.239.5 gw 10.42.3.1"; route "add
default gw 10.42.3.1"; };

}

root@client:~# <b>vtund -p home server</b>

root@client:~# <b>traceroute -n yahoo.com</b> traceroute to yahoo.com
(64.58.79.230), 30 hops max, 40 byte packets 1 208.201.239.32 24.368 ms
28.019 ms 19.114 ms 2 208.201.239.1 21.677 ms 22.644 ms 23.489 ms 3
208.201.224.194 20.41 ms 22.997 ms 23.788 ms 4 208.201.224.5 26.496
ms 23.8 ms 25.752 ms 5 206.24.221.217 26.174 ms 28.077 ms 26.344 ms 6
206.24.210.62 26.484 ms 27.851 ms 25.015 ms 7 206.24.226.103 104.22
ms 114.278 ms 108.575 ms 8 206.24.238.57 99.978 ms 99.028 ms 100.976
ms 9 206.24.238.26 103.749 ms 101.416 ms 101.09 ms 10 216.109.66.132
```

```
102.426 ms 104.222 ms 98.675 ms 11 216.33.98.19 99.985 ms 99.618 ms
103.827 ms 12 216.35.210.126 104.075 ms 103.247 ms 106.398 ms 13
64.58.77.41 107.219 ms 106.285 ms 101.169 ms

root@client:~# <b>ifconfig tun0</b> tun0 Link encap:Point-to-Point
Protocol inet addr:208.201.239.33 P-t-P:208.201.239.32
Mask:255.255.255.255

UP POINTOPOINT RUNNING MULTICAST MTU:1500 Metric:1

RX packets:39 errors:0 dropped:0 overruns:0 frame:0

TX packets:39 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:10

RX bytes:2220 (2.1 Kb) TX bytes:1560 (1.5 Kb)

root@client:~# <b>route</b> Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface 208.201.239.5
10.42.3.1 255.255.255.255 UGH 0 0 0 eth2

208.201.239.32 * 255.255.255.255 UH 0 0 0 tun0

10.42.3.0 * 255.255.255.192 U 0 0 0 eth2

10.42.4.0 * 255.255.255.192 U 0 0 0 eth0

loopback * 255.0.0.0 U 0 0 0 lo

default 208.201.239.32 0.0.0.0 UG 0 0 0 tun0

root@client:~# <b>ssh -f -N -c blowfish -C -L5000:localhost:5000
server</b> root@client:~# <b>vtund -p home localhost</b> root@client:~#
<b>traceroute -n yahoo.com</b> traceroute to yahoo.com (64.58.79.230),
30 hops max, 40 byte packets 1 208.201.239.32 24.715 ms 31.713 ms
29.519 ms 2 208.201.239.1 28.389 ms 36.247 ms 28.879 ms 3
208.201.224.194 48.777 ms 28.602 ms 44.024 ms 4 208.201.224.5 38.788
```

ms 35.608 ms 35.72 ms 5 206.24.221.217 37.729 ms 38.821 ms 43.489 ms 6 206.24.210.62 39.577 ms 43.784 ms 34.711 ms 7 206.24.226.103 110.761 ms 111.246 ms 117.15 ms 8 206.24.238.57 112.569 ms 113.2 ms 111.773 ms 9 206.24.238.26 111.466 ms 123.051 ms 118.58 ms 10 216.109.66.132 113.79 ms 119.143 ms 109.934 ms 11 216.33.98.19 111.948 ms 117.959 ms 122.269 ms 12 216.35.210.126 113.472 ms 111.129 ms 118.079 ms 13 64.58.77.41 110.923 ms 110.733 ms 115.22 ms

root@server:~# <b>iptables -A INPUT -t filter -i eth0 \</b> <b>-p tcp --dport 5000 -j DROP</b>

This allows local connections to get through (since they use loopback), and therefore requires an SSH tunnel to the server before accepting a connection.

As you can see, this can be an extremely handy tool to have around. In addition to giving live IP addresses to machines behind a NAT, you can effectively connect any two networks if you can obtain a single SSH connection between them (originating from either direction).

If your head is swimming from this *vtund.conf* configuration or you're feeling lazy and don't want to figure out what to change when setting up your own client's *vtund.conf* file, take a look at the automatic vtund.conf generator [Hack #79] .

Rob Flickenger (Linux Server Hacks)

```
# <b>ln -s vtundconf home</b> # <b>ln -s vtundconf tunnel2</b>

# <b>vtundconf home > /usr/local/etc/vtund.conf</b>

#!/usr/bin/perl -w
```

```
#

# vtund wrapper in need of a better name.
```

```
#

# (c)2002 Schuyler Erle & Rob Flickenger
```

#

################# CONFIGURATION

# If TunnelName is blank, the wrapper will look at @ARGV or $0.

```
#

# Config is TunnelName, LocalIP, RemoteIP, TunnelHost, TunnelPort,
Secret
```

```
#

my $TunnelName = "";

my $Config = q{

   home 208.201.239.33 208.201.239.32 208.201.239.5 5000 sHHH

   tunnel2 10.0.1.100 10.0.1.1 192.168.1.4 6001 foobar };
```

```
################ MAIN PROGRAM BEGINS HERE

use POSIX 'tmpnam';

use IO::File;

use File::Basename;

use strict;

# Where to find things...
```

```
#

$ENV{PATH} =
"/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/[RETURN]sbin"; my
$IP_Match = '((?:\d{1,3}\.){3}\d{1,3})'; # match xxx.xxx.xxx.xxx my
$Ifconfig = "ifconfig -a";

my $Netstat = "netstat -rn";

my $Vtund = "/bin/echo";

my $Debug = 1;

# Load the template from the data section.
```

```
#

my $template = join( "", );

# Open a temp file -- adapted from Perl Cookbook, 1st Ed., sec. 7.5.
```

```
#

my ( $file, $name ) = ("", "");

$name = tmpnam( )

   until $file = IO::File->new( $name, O_RDWR|O_CREAT|O_EXCL );
END { unlink( $name ) or warn "Can't remove temporary file $name!\n"; }

# If no TunnelName is specified, use the first thing on the command line, #
or if there isn't one, the basename of the script.

# This allows users to symlink different tunnel names to the same script.
```

```
#

$TunnelName ||= shift(@ARGV) || basename($0); die "Can't determine
tunnel config to use!\n" unless $TunnelName; # Parse config.
```

```
#

my ($LocalIP, $RemoteIP, $TunnelHost, $TunnelPort, $Secret); for
(split(/\r*\n+/, $Config)) {

   my ($conf, @vars) = grep( $_ ne "", split( /\s+/ )); next if not $conf or
$conf =~ /^\s*#/o; # skip blank lines, comments if ($conf eq $TunnelName)
{

   ($LocalIP, $RemoteIP, $TunnelHost, $TunnelPort, $Secret) = @vars; last;

   }

}

die "Can't determine configuration for TunnelName '$TunnelName'!\n"

   unless $RemoteIP and $TunnelHost and $TunnelPort; # Find the default
gateway.
```

```
#

my ( $GatewayIP, $ExternalDevice );

for (qx{ $Netstat }) {

   # In both Linux and BSD, the gateway is the next thing on the line, # and
the interface is the last.
#

   if ( /^(?:0.0.0.0|default)\s+(\S+)\s+.*?(\S+)\s*$/o ) {

   $GatewayIP = $1; $ExternalDevice = $2; last;

   }

}

die "Can't determine default gateway!\n" unless $GatewayIP and
$ExternalDevice; # Figure out the LocalIP and LocalNetwork.
```

```
#

my ( $LocalNetwork );

my ( $iface, $addr, $up, $network, $mask ) = ""; sub compute_netmask {

   ($addr, $mask) = @_; # We have to mask $addr with $mask because
linux /sbin/route # complains if the network address doesn't match the
netmask.
```

```
#

   my @ip = split( /\./, $addr ); my @mask = split( /\./, $mask ); $ip[$_] =
($ip[$_] + 0) & ($mask[$_] + 0) for (0..$#ip); $addr = join(".", @ip); return
$addr; }

for (qx{ $Ifconfig }) {

   last unless defined $_; # If we got a new device, stash the previous one (if
any).

   if ( /^([^\s:]+)/o ) {

   if ( $iface eq $ExternalDevice and $network and $up ) {

   $LocalNetwork = $network; last;

   }

   $iface = $1;

   $up = 0;

   }

   # Get the network mask for the current interface.

   if ( /addr:$IP_Match.*?mask:$IP_Match/io ) {

   # Linux style ifconfig.

   compute_netmask($1, $2); $network = "$addr netmask $mask"; } elsif (
/inet $IP_Match.*?mask 0x([a-f0-9]{8})/io ) {

   # BSD style ifconfig.

   ($addr, $mask) = ($1, $2); $mask = join(".", map( hex $_, $mask =~
/(..)/gs )); compute_netmask($addr, $mask); $network = "$addr/$mask"; }
```

```
    # Ignore interfaces that are loopback devices or aren't up.

    $iface = "" if /\bLOOPBACK\b/o; $up++ if /\bUP\b/o; }

die "Can't determine local IP address!\n" unless $LocalIP and
$LocalNetwork; # Set OS dependent variables.
```

```
#

my ( $GW, $NET, $PTP );

if ( $^O eq "linux" ) {

   $GW = "gw"; $PTP = "pointopoint"; $NET = "-net"; } else {

   $GW = $PTP = $NET = ""; }

# Parse the config template.
```

```
#

$template =~ s/(\$\w+)/$1/gee;

# Write the temp file and execute vtund.
```

```
#

if ($Debug) {

   print $template; } else {

   print $file $template; close $file;

   system("$Vtund $name"); }

_ _DATA_ _

options {

   port $TunnelPort; ifconfig /sbin/ifconfig; route /sbin/route; }

default {

   compress no;

   speed 0;

}

# 'mytunnel' should really be `basename $0` or some such # for automagic
config selection

$TunnelName {

   type tun;

   proto tcp;

   keepalive yes; pass $Secret; up {

   ifconfig "%% $LocalIP $PTP $RemoteIP arp"; route "add $TunnelHost
$GW $GatewayIP"; route "delete default"; route "add default $GW
$RemoteIP"; route "add $NET $LocalNetwork $GW $GatewayIP"; };
```

```
  down {

  ifconfig "%% down"; route "delete default"; route "delete $TunnelHost
$GW $GatewayIP"; route "delete $NET $LocalNetwork"; route "add
default $GW $GatewayIP"; };

}
```

Rob Flickenger (Linux Server Hacks)

```
$ <b>tar xfz openvpn-1.5.0.tar.gz</b> $ <b>cd openvpn-1.5.0</b> $
<b>./configure && make</b>
```

```
$ <b>./configure --with-lzo-headers=/usr/local/include \</b> <b>--with-lzo-
lib=/usr/local/lib</b>
```

```
LZO library and headers not found.

LZO library available from http://www.oberhumer.com/opensource/lzo/

configure: error: Or try ./configure --disable-lzo
```

```
configure: checking for LZO Library and Header files...

checking lzo1x.h usability... yes checking lzo1x.h presence... yes checking
for lzo1x.h... yes

checking for lzo1x_1_15_compress in -llzo... yes
```

```
# <b>openvpn --remote zul --dev tun0 --ifconfig 10.0.0.19 10.0.0.5</b>
```

```
# <b>openvpn --remote kryten --ifconfig 10.0.0.5 10.0.0.19</b>
```

```
[andrew@kryten andrew]$ <b>/sbin/ifconfig tun0</b> tun0:
flags=51<UP,POINTOPOINT,RUNNING> mtu 1300

   inet 10.0.0.19 --> 10.0.0.5 netmask 0xffffffff
```

```
[andrew@kryten andrew]$ <b>ping -c 4 10.0.0.5</b> PING 10.0.0.5
(10.0.0.5): 56 data bytes 64 bytes from 10.0.0.5: icmp_seq=0 ttl=255
time=0.864 ms 64 bytes from 10.0.0.5: icmp_seq=1 ttl=255 time=1.012 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=255 time=0.776 ms 64 bytes from
10.0.0.5: icmp_seq=3 ttl=255 time=0.825 ms --- 10.0.0.5 ping statistics ---

4 packets transmitted, 4 packets received, 0% packet loss round-trip
min/avg/max = 0.776/0.869/1.012 ms
```

```
dev tun0
```

```
ifconfig 10.0.0.5 10.0.0.19

up /etc/openvpn/openvpn.up

down /etc/openvpn/openvpn.down tls-server

dh /etc/openvpn/dh1024.pem

ca /etc/ssl/ca.crt

cert /etc/ssl/zul.crt

key /etc/ssl/private/zul.key

ping 15

verb 0

# <b>openssl dhparam -out dh1024.pem 1024</b>

dev tun0
```

```
remote zul

ifconfig 10.0.0.19 10.0.0.5

up /etc/openvpn/openvpn.up

down /etc/openvpn/openvpn.down tls-client

ca /etc/ssl/ca.crt

cert /etc/ssl/kryten.crt

key /etc/ssl/private/kryten.key ping 15

verb 0

#!/bin/sh

/sbin/route add -net 10.0.0.0 gw $5 netmask 255.255.255.0

#!/bin/sh

arp -s $5 00:00:d1:1f:3f:f1 permanent pub

#!/bin/sh

arp -d kryten

# <b>openvpn --config /etc/openvpn/openvpn.conf --daemon</b>
```

Setting up OpenVPN under Windows is even easier. Simply run the installer, and everything you need will be installed onto your system. This includes OpenSSL, the TUN/TAP driver, and OpenVPN itself. The installer will also associate the *.ovpn* file extension with OpenVPN. Simply put your configuration information in a *.ovpn* file, double-click it, and you're ready to go.

This should get you started using OpenVPN, but it has far too many configuration options to discuss here. Be sure to look at the OpenVPN web site for more information.

```
# <b>/usr/sbin/pppd updetach noauth silent nodeflate \ </b> <b>pty
"/usr/bin/ssh root@colossus /usr/sbin/pppd nodetach notty noauth" \ </b>
<b>ipparam 10.1.1.20:10.1.1.1</b> root@colossus's password:

local IP address 10.1.1.20

remote IP address 10.1.1.1

$ <b>/sbin/ifconfig ppp0</b>

ppp0 Link encap:Point-to-Point Protocol

   inet addr:10.1.1.20 P-t-P:10.1.1.1 Mask:255.255.255.255

   UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500
Metric:1

   RX packets:58 errors:0 dropped:0 overruns:0 frame:0

   TX packets:50 errors:0 dropped:0 overruns:0 carrier:0

   collisions:0 txqueuelen:3

   RX bytes:5372 (5.2 Kb) TX bytes:6131 (5.9 Kb)

$ <b>ping 10.1.1.1</b>

PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.

64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=4.56 ms 64 bytes from
10.1.1.1: icmp_seq=2 ttl=64 time=4.53 ms 64 bytes from 10.1.1.1:
icmp_seq=3 ttl=64 time=5.45 ms 64 bytes from 10.1.1.1: icmp_seq=4
ttl=64 time=4.51 ms --- 10.1.1.1 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3025ms rtt
min/avg/max/mdev = 4.511/4.765/5.451/0.399 ms

$ <b>ssh 10.1.1.1</b>
```

```
The authenticity of host '10.1.1.1 (10.1.1.1)' can't be established.

RSA key fingerprint is 56:36:db:7a:02:8b:05:b2:4d:d4:d1:24:e9:4f:35:49.

Are you sure you want to continue connecting (yes/no)? yes Warning:
Permanently added '10.1.1.1' (RSA) to the list of known hosts.

andrew@10.1.1.1's password:

[andrew@colossus andrew]$
```

Before deciding to keep this setup, you may want to generate login keys to use with `ssh` [Hack #73], so that you don't need to type in a password each time. In addition, you may want to create a separate user for logging in on the remote machine and starting `pppd`. However, `pppd` needs to be started as root, so you'll have to make use of sudo [Hack #6]. Also, you can enable SSH's built-in compression by adding a `-C` to the `ssh` command. In some circumstances, SSH compression can greatly improve the speed of the link. Finally, to tear down the tunnel, simply kill the `ssh` process that `pppd` spawned.

Although it's ugly and might not be as stable and full of features as actual VPN implementations, the PPP and SSH combination can help you create an instant encrypted network without the need to install additional software.

See Also

- The section "Creating a VPN with PPP and SSH" in Virtual Private Networks, Second Edition, by Charlie Scott, Paul Wolfe, and Mike Erwin (O'Reilly)

Chapter 7. Network Intrusion Detection

Hacks #82-95

One class of tools that's come to the forefront in network security in recent years is network

intrusion detection systems (NIDS). These systems can be deployed on your network and monitor the traffic until they detect suspicious behavior, when they spring into action and notify you of what is going on. They are excellent tools to use in addition to your logs, since a network IDS can often spot an attack before it reaches the intended target or has a chance to end up in your logs.

Currently, there are two main types of NIDS. The first type detects intrusions by monitoring the traffic for specific byte patterns that are similar to known attacks. A NIDS that operates in this manner is known as a signature-based intrusion detection system. The other type of network IDS is a statistical monitor. These monitor the traffic on the network, but instead of looking for a particular pattern or signature, they maintain a statistical history of the packets that pass through your network, and report when they see a packet that falls outside of the normal network traffic pattern. NIDS that employ this method are known as *anomaly-based*

intrusion detection systems.

In this chapter you'll learn how to set up Snort, a signature-based IDS. You'll also learn how to set up

Snort with SPADE, which adds anomaly-detection capabilities to Snort, giving you the best of both worlds. This chapter also demonstrates how to set up several different applications that can help you to monitor and manage your NIDS once you have it deployed.

Finally, you'll see how to set up a system that appears vulnerable to attackers, but is actually quietly waiting and monitoring everything it sees. These systems are called *honeypots*, and the last few hacks will show

you how to quickly and easily set one up, and how to monitor intruders that have been fooled and trapped by it.

```
$ <b>./configure && make</b>

# <b>make install</b>

<b>$ ./configure --with-libpcap-includes=/opt/include\</b>

<b> --with-libpcap-libraries=/opt/lib</b>

# <b>./snort -evi eth0</b>
```

```
Running in packet dump mode

Log directory = /var/log/snort

Initializing Network Interface eth0

    --== Initializing Snort ==--

Initializing Output Plugins!

Decoding Ethernet on interface eth0

    --== Initialization Complete ==--

-*> Snort! <*-

Version 2.0.5 (Build 98)

By Martin Roesch (roesch@sourcefire.com, www.snort.org)
```

```
12/14-16:25:17.874711 0:A:95:C7:2B:10 -> 0:C:29:E2:2B:C1 type:0x800
len:0x42

192.168.0.60:53179 -> 192.168.0.41:22 TCP TTL:64 TOS:0x10 ID:56177
IpLen:20 DgmLen:52 DF

***A**** Seq: 0x67E53951 Ack: 0x2BA09FF7 Win: 0xFFFF TcpLen: 32

TCP Options (3) => NOP NOP TS: 3426501948 469087

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+=+=+=+=+=+=+=+

12/14-16:25:17.874828 0:C:29:E2:2B:C1 -> 0:A:95:C7:2B:10 type:0x800
len:0x252

192.168.0.41:22 -> 192.168.0.60:53179 TCP TTL:64 TOS:0x10 ID:50923
IpLen:20 DgmLen:580 DF

***AP*** Seq: 0x2BA09FF7 Ack: 0x67E53951 Win: 0x2200 TcpLen: 32

TCP Options (3) => NOP NOP TS: 469100 3426501948
```

=+

```
# <b>mkdir /usr/local/etc/snort &&\</b>

<b> cp etc/[^Makefile]* /usr/local/etc/snort</b>

var HOME_NET any



var EXTERNAL_NET any

var HOME_NET [10.1.1.0/24,192.168.1.0/24]

output database: log, mysql, user=snort password=snortpass
dbname=SNORT \ host=dbserver

output database: alert mysql, user=snort password=snortpass
dbname=SNORT \ host=dbserver

# <b>mysql SNORT -p < ./contrib/create_mysql</b>

# <b>snort -T -c /usr/local/etc/snort/snort.conf</b>

# <b>snort -Dd -z est -c /usr/local/etc/snort/snort.conf</b>
```

Two of these flags, `-d` and `-c`, were used previously (to tell Snort to decode packet data and to use the specified configuration file, respectively). The other two are new. The `-D` flag tells Snort to print out some startup messages and then fork into the background. The `-z est` argument tells Snort's streams preprocessor plug-in to ignore TCP packets that aren't part of established sessions, which makes your Snort system much less susceptible to spoofing attacks and certain DoS attacks. Some other useful options are `-u` and `-g`, which let Snort drop its privileges and run under the user and group that you specify. These are especially useful with the `-t` option, which will `chroot()` Snort to the directory that you specify. Now you should start to see logs appearing in */var/log/snort*.

See Also

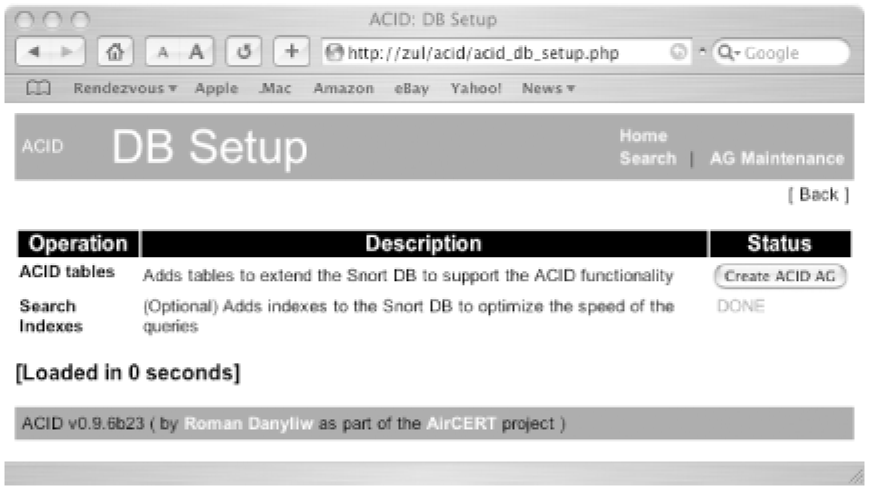
- Chapter 11, "Simple Intrusion Detection Techniques," in Building Secure Servers with Linux, by Michael D. Bauer (O'Reilly)

```
$Dblib_path = "../adodb";


$Dbtype = "mysql";


$alert_dbname = "SNORT";


$alert_host = "localhost";


$alert_port = "";


$alert_user="snort";


$alert_password = "snortpass";
$archive_dbname


$archive_host


$archive_port


$archive_user
```

```
$archive_password

$ChartLib_path = "../jpgraph-1.13/src";
```

Congratulations! You're finished mucking about in configuration files for the time being. Now open a web browser and go to the URL that corresponds to the directory where you unpacked *ACID*. You should then be greeted with a database setup page as shown in Figure 7-1.

Figure 7-1. The ACID database setup page



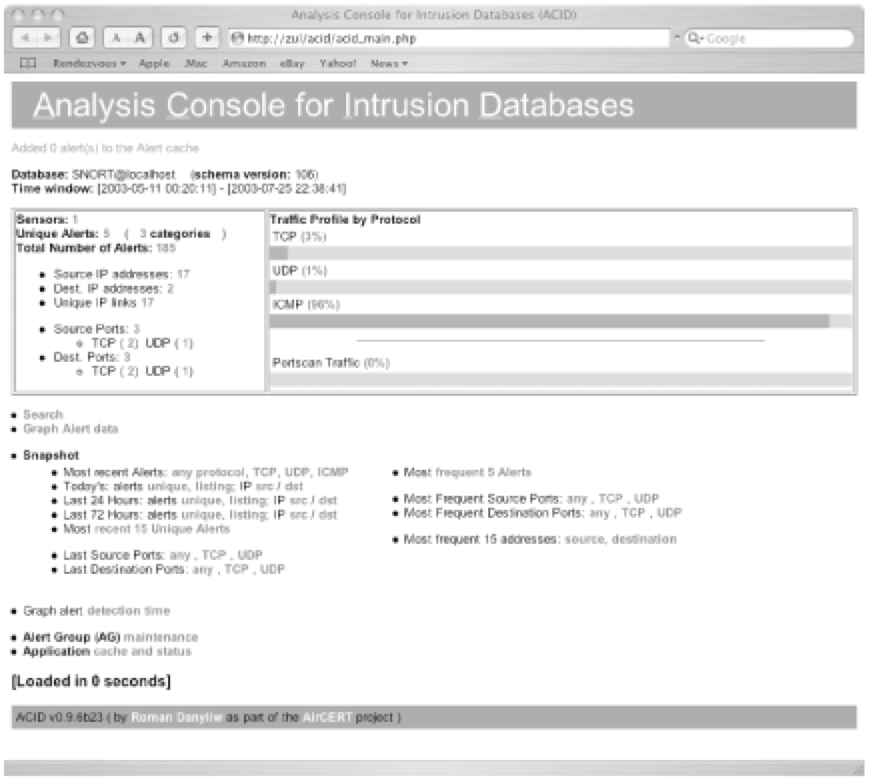
Operation	Description	Status
ACID tables	Adds tables to extend the Snort DB to support the ACID functionality	Create ACID AG
Search Indexes	(Optional) Adds indexes to the Snort DB to optimize the speed of the queries	DONE

[Loaded in 0 seconds]

ACID v0.9.6b23 (by Roman Danyliw as part of the AirCERT project)

Before you can use *ACID*, it must create some database tables for its own use. To do this, click the Create ACID AG button. After this, you should see a screen confirming that the tables were created. In addition, you can have *ACID* create indexes for your events table if this was not done prior to setting up *ACID*. Indexes will greatly speed up queries as your events table grows, at the expense of using a little more disk space. Once you are done with the setup screen, you can click the Home link to go to the main *ACID* page, as seen in Figure 7-2.

Figure 7-2. ACID's main page


Analysis Console for Intrusion Databases (ACID)
http://zul/acid/acid_main.php
Google
Rendezvous
Apple
.Mac
Amazon
eBay
Yahoo!
News

Analysis Console for Intrusion Databases

Added 0 alert(s) to the Alert cache

Database: SNORT@localhost (**schema version:** 106)
Time window: [2003-05-11 00:20:11] - [2003-07-25 22:38:41]

Sensors: 1
Unique Alerts: 5 (3 **categories**)
Total Number of Alerts: 185

- Source IP addresses: 17
- Dest. IP addresses: 2
- Unique IP links 17
- Source Ports: 3
  - TCP (2) UDP (1)
- Dest. Ports: 3
  - TCP (2) UDP (1)

Traffic Profile by Protocol
TCP (3%)
UDP (1%)
ICMP (96%)
Portscan Traffic (0%)

- Search
- Graph Alert data
- **Snapshot**
  - Most recent Alerts: any protocol, TCP, UDP, ICMP
  - Today's: alerts unique, listing; IP src / dst
  - Last 24 Hours: alerts unique, listing; IP src / dst
  - Last 72 Hours: alerts unique, listing; IP src / dst
  - Most recent 15 Unique Alerts
  - Last Source Ports: any , TCP , UDP
  - Last Destination Ports: any , TCP , UDP
  - Most frequent 5 Alerts
  - Most Frequent Source Ports: any , TCP , UDP
  - Most Frequent Destination Ports: any , TCP , UDP
  - Most frequent 15 addresses: source, destination
- Graph alert detection time
- **Alert Group (AG)** maintenance
- **Application** cache and status

[Loaded in 0 seconds]

ACID v0.9.6b23 (by Roman Danyliw as part of the AirCERT project)

ACID has a fairly intuitive user interface. The main table provides plenty of links to see many useful views of the database at a glance, such as the list of source or destination IP addresses associated with the alerts in your database, as well as the source and destination ports.

```
$ <b>mysql -u root -p</b> Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 546 to server version: 3.23.55

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> <b>CREATE DATABASE SGUIL;</b> Query OK, 1 row affected
(0.00 sec) mysql> <b>GRANT SELECT,INSERT,UPDATE,DELETE ON
SGUIL.* \</b> <b> TO sguil IDENTIFIED BY 'sguilpass';</b> Query OK,
0 rows affected (0.06 sec) mysql> <b>FLUSH PRIVILEGES;</b> Query
OK, 0 rows affected (0.06 sec) mysql>

$ <b>mysql -u root -p SGUIL < create_sguildb.sql</b>

$ <b>tcl</b>

tcl><b>package require Tclx</b>
```

8.3

```
tcl><b>package require mysqltcl</b>
```

2.40

tcl>

$ <b>tcl</b>

tcl><b>package require tls</b>

1.41

tcl>

# <b>mkdir /etc/sguild</b> # <b>cd server</b>

# <b>cp autocat.conf sguild.conf sguild.queries \</b> <b> sguild.users /etc/sguild</b>

set DBNAME SGUIL

set DBPASS sguilpass

set DBHOST localhost

set DBPORT 3389

set DBUSER sguil

set RULESDIR /etc/snort/rules

# <b>sguild -adduser andrew</b> Please enter a passwd for andrew: Retype passwd:

User 'andrew' added successfully

set OPENSSL 1

set TLS_PATH /usr/lib/tls1.4/libtls1.4.so

$ <b>cd ~/snort-2.0.5/src/preprocessors</b> $ <b>patch spp_portscan.c < \ </b> <b> ~/sguil0.3.0/sensor/snort_mods/2_0/spp_portscan_sguil.patch</b> patching file spp_portscan.c

$ <b>patch spp_stream4.c < \</b> <b> ~/sguil-0.3.0/sensor/snort_mods/2_0/spp_stream4_sguil.patch</b> patching file

```
spp_stream4.c
```

```
Hunk #9 succeeded at 988 (offset -5 lines).
```

```
Hunk #11 succeeded at 3324 (offset -5 lines).
```

```
Hunk #13 succeeded at 3674 (offset -5 lines).
```

```
preprocessor portscan: $HOME_NET 4 3 /var/log/snort/portscans gw-ext0
```

```
preprocessor stream4: detect_scans, disable_evasion_alerts, keepstats db \
/var/log/snort/ssn_logs
```

```
output alert_unified: filename snort.alert, limit 128
```

```
output log_unified: filnemae snort.log, limit 128
```

```
00 0-23/1 * * * /usr/local/bin/log_packets.sh restart
```

```
$ <b>cd ~/barnyard-0.1.0/src/output-plugins</b> $ <b>cp ~/sguil-
0.3.0/sensor/barnyard_mods/op_sguil.* .</b>
```

```
#include "op_acid_db.h"
```

```
#include "op_acid_db.h"
```

```
#include "op_sguil.h"
```

```
AcidDbOpInit( );
```

```
AcidDbOpInit( );
```

```
SguilOpInit( );
```

```
output sguil: mysql, sensor_id 0, database SGUIL, server localhost, user
sguil, sguilpass, sguild_host localhost, sguild_port 7736
```

```
set SERVER_HOST localhost
```

set SERVER_PORT 7736

set HOSTNAME gw-ext0

set PORTSCAN_DIR /var/log/snort/portscans set SSN_DIR /var/log/snort/ssn_logs set WATCH_DIR /var/log/snort

set LOCALSENSOR 1

set LOCAL_LOG_DIR /var/log/snort/archive set REMOTE_LOG_DIR /var/log/snort/dailylogs

# <b>sguild -O /usr/lib/tls1.4/libtls1.4.so</b> # <b>xscriptd -O /usr/lib/tls1.4/libtls1.4.so</b>

If you're not using SSL, you should omit the `-O /usr/lib/tls1.4/libtls1.4.so` portions of the commands. Otherwise, you should make sure that the argument to `-O` points to the location of `libtls` on your system.

Getting *Sguil* running isn't trivial, but it is well worth the effort. Once everything is running, you will have a very good overview of precisely what is happening on your network. *Sguil* presents data from a bunch of sources simultaneously, giving you a good view of the big picture that is sometimes impossible to see when simply looking at your NIDS logs.

```
# <b>tar xfz snortcenter-v1.0-RC1.tar.gz</b> # <b>cp -R www
/var/www/htdocs/snortcenter</b>
```

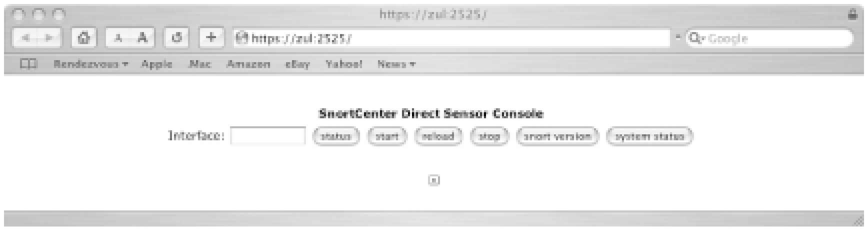
```
$DBlib_path = "../adodb/";

$DBtype = "mysql";

$DB_dbname = "SNORTCENTER";

$DB_host = "localhost";

$DB_port = "";

$DB_user = "snortcenter";

$DB_password = "snortcenterpass";

$hidden_key_num =1823701983719312;
```

```
$ <b>mysql -u root -p mysql</b> Enter password:

Reading table information for completion of table and column names You
can turn off this feature to get a quicker startup with -A Welcome to the
MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 27 to server version: 3.23.55

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> <b>create database SNORTCENTER;</b> Query OK, 1 row
affected (0.01 sec) mysql> <b>GRANT
SELECT,INSERT,UPDATE,DELETE ON SNORTCENTER.* TO </b>\
<b>snortcenter@localhost IDENTIFIED BY 'snortcenterpass';</b> Query
OK, 0 rows affected (0.00 sec) mysql> <b>FLUSH PRIVILEGES;</b>
Query OK, 0 rows affected (0.02 sec) mysql> Bye

$ <b>mysql -u root -p SNORTCENTER < snortcenter_db.mysql</b>
```

```
# <b>perl -MCPAN -e "install Net::SSLeay"</b>

# <b>tar xfz /tmp/snortcenter-agent-v1.0-RC1.tar.gz</b> # <b>cp -R sensor
/usr/local/snortcenter</b>

# <b>cd /usr/local/snortcenter</b> # <b>mkdir conf</b>

# <b>openssl req -new -x509 -days 3650 -nodes \</b> <b> -out
conf/sensor.pem -keyout conf/sensor.pem</b>

# <b>sh setup.sh</b>

***************************************************************
***************

* Welcome to the SnortCenter Sensor Agent setup script, version 1.0 RC1 *

***************************************************************
***************

Installing Sensor in /usr/local/snortcenter ...

***************************************************************
***************

The Sensor Agent uses separate directories for configuration files and log
files.

Unless you want to place them in a other directory, you can just accept the
defaults.

Config file directory [/usr/local/snortcenter/conf]:
```

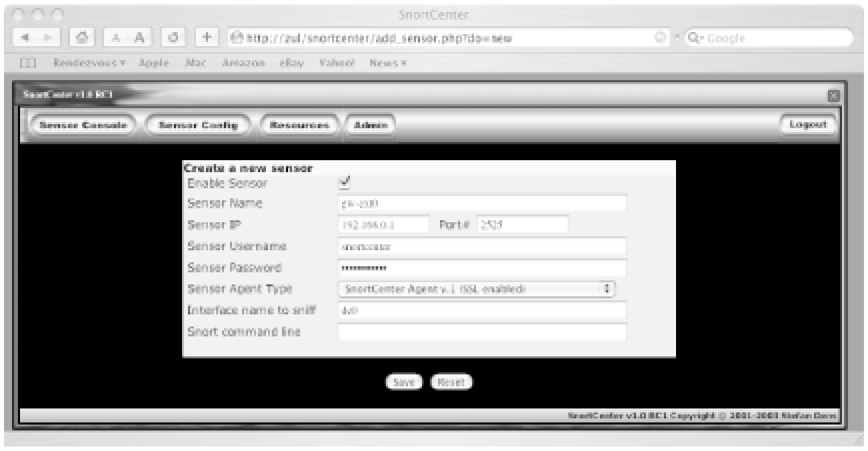
This script will prompt you for several pieces of information, such as the sensor agent's configuration file and log directories, the full path to the `perl` binary (e.g., */usr/bin/perl*), as well as the location of your `snort` binary and rules. In addition, it will ask you questions about your operating system, what port and IP address you want the sensor agent to listen on (the default is TCP port 2525), and what IP addresses are allowed to connect to the agent. In particular, it will ask you to set a login and password that the management console will use for logging into the agent. After it has prompted you for all the information it needs, it will start the sensor agent on the port and IP address specified in the configuration file. You can now test out the sensor agent by accessing it with your web browser (be sure to use *https* instead of *http*). You should see a page similar to Figure 7-9 after entering the login information contained in the setup script.

Figure 7-9. The sensor agent direct console page



Now you can go back to the main management console and add the sensor to it. To do this, log back into the management console and select Add Sensor from the Sensor Console menu. After doing this, you should see something similar to Figure 7-10.

Figure 7-10. Adding a sensor agent

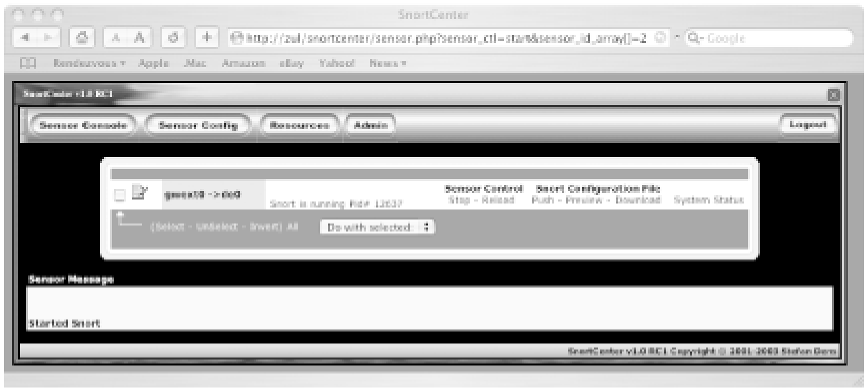


Create a new sensor

Enable Sensor	✓
Sensor Name	gw-zul0
Sensor IP	192.168.0.1 Port# 2525
Sensor Username	snortcenter
Sensor Password	•••••••••••
Sensor Agent Type	SnortCenter Agent v.1 (SSL enabled)
Interface name to sniff	de0
Snort command line	

Fill in the information that you used when running the setup script and click the Save button. When the next page loads, the sensor that you just added should appear in the sensor list. You can push a basic configuration to the sensor by opening the Admin menu, then selecting the Import/Update Rules item, and then Update from Internet. After you've done that, go back to the sensor list by clicking View Sensors in the Sensor Consoles menu, and then click the Push hyperlink for the sensor. To start Snort on that particular sensor, click the Start link. After you've done that, you should see a page similar to Figure 7-11.

Figure 7-11. SnortCenter's sensor list after starting a sensor


SnortCenter
http://2ul/snortcenter/sensor.php?sensor_ctl=start&sensor_id_array[]=2
Rendezvous Apple .Mac Amazon eBay Yahoo! News
Sensor Console
Sensor Config
Resources
Admin
Logout
Sensor Control
Stop - Reload
Snort Configuration File
Push - Preview - Download
System Status
(Select - UnSelect - Invert) All
Do with selected:
Sensor Message

gwex10 -> dc0

Snort is running Pid# 12637

Started Snort

SnortCenter v1.0 RC1 Copyright © 2001-2003 Stefan Dens

You can now configure your sensor by using the Sensor Config and Resources menus. Once you've created a configuration you're satisfied with, you can push it to your sensor(s) by going back to the sensor list and selecting Push.

<tt><i>action</i></tt> <tt><i>proto</i></tt> <tt><i>src_ip</i></tt> <tt><i>src_port</i></tt> <tt><i>direction</i></tt> <tt><i>dst_ip</i></tt> <tt><i>dst_port</i></tt> (<tt><i>options</i></tt>)

```
activate tcp any any -> 192.168.1.21 22 (content:"/bin/sh"; activates:1; \

msg:"Possible SSH buffer overflow"; )

dynamic tcp any any -> 192.168.1.21 22 (activated_by:1; count:100;)
```

```
ruletype redalert


{



   type alert



   output alert_syslog: LOG_AUTH LOG_ALERT



   output database: log, mysql, user=snort dbname=snort host=localhost

}
```

[192.168.1.1,192.168.1.45,10.1.1.24]

1:1024

1024:

:1024

```
alert tcp 192.168.1.35 any -> any any (msg:"Traffic from 192.168.1.35";)
```

```
alert tcp any any -> any any (msg:"Possible exploit"; content:"|90|";)
```

```
alert tcp any any -> any any (msg:"Possible exploit"; content:"|90|"; \

offset:40; depth:75;)
```

```
alert tcp any any -> any any (msg:"Possible exploit"; content:"|90|"; \

offset:40; depth:75; dsize: >6000;)
```

```
alert any any -> any any (flags: SF,12; msg: "Possible SYN FIN scan";)
```

Valid flags are `S` for SYN, `F` for FIN, `R` for RST, `P` for PSH, `A` for ACK, and `U` for URG. In addition, Snort lets you check the values of the two reserved flag bits. You can specify these by using either `1` or `2`. You can also match packets that have no flags set by using `0`. There are also several operators that the `flags` option will accept. You can prepend either a `+` , `*`, or `!` to the flags, to match on all the flags plus any others, any of the flags, or only if none of the flags are set, respectively.

One of the best features of Snort is that it provides many plug-ins that can be used in the options field of a rule. The options discussed here should get you off to a good start. However, if you want to write more complex rules, consult Snort's excellent rule documentation, which contains full descriptions and examples for each of Snort's rule options. The Snort User's Manual is available at http://www.snort.org/docs/writing_rules/.

```
$ <b>locate ip_queue.o</b> /usr/src/linux-2.4.20-
8/net/ipv4/netfilter/ip_queue.o /usr/src/linux-2.4.20-
8/net/ipv4/netfilter/.ip_queue.o.flags /lib/modules/2.4.20-
8/kernel/net/ipv4/netfilter/ip_queue.o

$ <b>locate libipq</b>

/usr/include/libipq.h

/lib/libipq.a

$ <b>./configure --enable-inline && make</b>

gcc -DHAVE_CONFIG_H -I. -I. -I../.. -I../.. -I../../src -I../../src/sfutil -
I/usr/include/pcap -I../../src/output-plugins -I../../src/detection-plugins -
I../../src/

preprocessors -I../../src/preprocessors/flow -I../../src/preprocessors/portscan
-I../../

src/preprocessors/flow/int-snort -I../../src/preprocessors/HttpInspect/include
-I/usr/

include/pcre -I/usr/local/include -g -O2 -Wall -DGIDS -D_BSD_SOURCE
-D__BSD_SOURCE -D_

_FAVOR_BSD -DHAVE_NET_ETHERNET_H -
DLIBNET_LIL_ENDIAN -c `test -f 'spo_alert_fast.c' ||

echo './'`spo_alert_fast.c

In file included from /usr/include/linux/netfilter_ipv4/ip_queue.h:10, from
/usr/include/libipq.h:37, from ../../src/inline.h:8, from ../../src/snort.h:38,
from spo_alert_fast.c:51: /usr/include/linux/if.h:59: redefinition of `struct
ifmap'

/usr/include/linux/if.h:77: redefinition of `struct ifreq'
```

```
/usr/include/linux/if.h:126: redefinition of `struct ifconf'

make[3]: *** [spo_alert_fast.o] Error 1

make[3]: Leaving directory `/home/andrew/snort_inline-2.1.0/src/output-plugins'

make[2]: *** [all-recursive] Error 1

make[2]: Leaving directory `/home/andrew/snort_inline-2.1.0/src'

make[1]: *** [all-recursive] Error 1

make[1]: Leaving directory `/home/andrew/snort_inline-2.1.0'

make: *** [all] Error 2
```

```
#include <linux/if.h>

#include <net/if.h>
```

```
$ <b>make clean && make</b>
```

```
# <b>iptables -F</b>

# <b>iptables -A INPUT -j QUEUE</b> # <b>iptables -A OUTPUT -j QUEUE</b> # <b>iptables -A FORWARD -j QUEUE</b>

# <b>snort_inline -Qvc /etc/snort/snort_inline.conf</b>
```

If you're administering the machine remotely, you'll probably want to start `snort_inline` before enabling the `QUEUE` targets, since it's `snort_inline` that will actually pass the packets back and forth. Otherwise, your remote logins will be dropped as soon as you put the iptables rules in place. If you're particularly paranoid, have your `QUEUE` target rules ignore packets coming from a certain IP address or range of addresses.

```
$ <b>sh makesnortsam.sh</b>

$ <b>tar xvfz snortsam-patch.tar.gz </b>
```

NOTE

patchsnort.sh

patchsnort.sh.asc

snortpatch8

snortpatch8.asc

snortpatch9

snortpatch9.asc

snortpatchb

snortpatchb.asc

```
$ <b>sh patchsnort.sh snort-2.0.5</b>
Patching Snort version 2.0...
```

```
patching file spo_alert_fwsam.c

patching file spo_alert_fwsam.h

patching file twofish.c

patching file twofish.h

patching file plugbase.c

Hunk #1 succeeded at 29 with fuzz 2 (offset -73 lines).


Hunk #2 succeeded at 639 with fuzz 2 (offset 77 lines).


Patching Makefiles...


Done

accept 192.168.1.0/24, qwijybo

bindip 192.168.1.15

sam_server port 1813

pix 192.16.1.2 <tt><i>telnetpw enablepw</i></tt>

output alert_fwsam: firewall/mypassword firewall2:1025/mypassword

fwsam: src, 5 minutes;

# <b>snortsam /usr/local/etc/snortsam.conf</b>
```

Of course, you'll need to substitute the full path to your configuration file if it's not */usr/local/etc/snortsam.conf*. As for Snort, just start it as you normally would.

For more information on using *SnortSam* with other types of firewalls, be sure to check out the *README* files included with the source distribution.

```
$ <b>make SNORTBASE=../snort-2.0.5</b>
```

```
var SPADEDIR /var/log/snort/spade
```

```
peprocessor spade-homenet: 192.168.1.0/24
```

```
include spade.conf
```

Run Snort just as you did before. *Spade* will now send its output to any of the output plug-ins that you have configured when it detects anomalous behavior. This is triggered when a given packet's anomaly score is in the range of .8 to .9 (it depends on the type of packet). Any alerts generated by *Spade* will be prefixed with `Spade`: and will include a description of the packet's deviant behavior and its anomaly score.

# alert icmp any any -> any any (msg:"ICMP Destination Unreachable (Communication Administratively Prohibited)"; itype: 3; icode: 13; sid:485; classtype:misc-activity; rev:2;)

disablesid 485

# <b>oinkmaster.pl -o /etc/snort/rules</b>

Now you won't have to remember which rules to disable ever again.

```
# <b>tcpdump -i eth1</b>

tcpdump: bind: Network is down

# <b>ifconfig eth1 up promisc</b> # <b>ifconfig eth1</b>

eth1 Link encap:Ethernet HWaddr 00:DE:AD:BE:EF:00

   UP BROADCAST PROMISC MULTICAST MTU:1500 Metric:1

   RX packets:0 errors:0 dropped:0 overruns:0 frame:0

   TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

   collisions:0 txqueuelen:100

   RX bytes:0 (0.0 b) TX bytes:0 (0.0 b) Interrupt:11 Base address:0x1c80

# <b>/usr/sbin/tcpdump -i eth1</b> tcpdump: WARNING: eth1: no IPv4
address assigned tcpdump: listening on eth1
```

After you've put the interface up, just start your IDS [Hack #82] . Your IDS will run as normal, but since there is no way to directly access the machine, it is very difficult to attack it.

However, just like potential attackers, you will be unable to access the machine remotely. Therefore, if you want to manage the sensor remotely, you'll need to put in a second network interface. Of course, if you did this and hooked it up to the same network that the IDS sensor is monitoring, it would totally defeat the purpose of running the other interface without an IP address. To keep the traffic isolated, you should create a separate network for managing the IDS sensors. You can of course attach this network to one that is remotely accessible and then firewall it heavily.

Another approach is to access the box using an alternate channel, such as a serial port connected to another machine that does have a network connection. Just run a console on the serial port, and take care to heavily secure the second machine. You could also connect a modem (remember those?) to an unlisted phone number or, better yet, an unlisted extension on your office's PBX. Depending on your situation, simply using the console for access may be the simplest and most secure method.

Whichever method you decide to use for remote access is a choice you'll have to make by weighing the value of increased security against the inconvenience of jumping through hoops to access the machine. Security nearly always involves a trade-off between convenience and confidence.

```
$ <b>./configure --enable-mysql</b>
```

```
output alert_unified: filename snort.alert, limit 128
```

```
output log_unified: filnemae snort.log, limit 128
```

```
config daemon

config hostname: colossus



config interface: eth0

processor dp_alert

processor dp_log

output alert_fast: fast_alerts.log

output log_dump: ascii_dump.log

output alert_syslog: LOG_AUTH LOG_ALERT

output alert_syslog: hostname=loghost, LOG_AUTH LOG_ALERT

output log_pcap: alerts.pcap

output alert_acid_db: mysql, sensor_id 0, database SNORT, server dbserver,
user snort

output log_acid_db: mysql, database SNORT, server dbserver, user snort,
detail full
```

```
# <b>barnyard -f snort.alert</b>
```

```
# <b>barnyard -c /usr/local/etc/snort/barnyard.conf \ </b>

<b>-g /usr/local/etc/snort/gen-msg.map \</b>

<b>-s /usr/local/etc/snort/sid-msg.map -f snort.alert</b>
```

This would tell *Barnyard* where to find all the files it needs if they are in */usr/local/etc/snort* (and are too stubborn to create a symlink to */etc/snort*).

If you're using a directory other than */var/log/snort* to store Snort's logs, you can specify it with the `-d` option.

Congratulations. With *Barnyard* running, you should be able to handle much larger volumes of traffic without dropping log entries or missing a single packet.

```
# <b>apxs -cia mod_security.c</b>

--activate-module=src/modules/extra/mod_security

--enable-module=security

<IfModule mod_security.c>

   SecFilterEngine On

</IfModule>
```

```
SecFilterScanPOST On
```

```
SecFilterCheckURLEncoding On
```

```
SecFilterCheckUnicodeEncoding On

SecFilterForceByteRange 32 126

SecFilter "\.\./"

SecFilter "<[[:space:]]*script"

SecFilter "<(.|\n)+>"

SecFilter "delete[[:space:]]+from"


SecFilter "insert[[:space:]]+into"


SecFilter "select.+from"

SecFilterSelective COOKIE_sessionid "!^(|[0-9]{1,9})$"

SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"

SecFilterSelective "HTTP_CONTENT_TYPE" multipart/form-data

SecFilterSelective "HTTP_ACCEPT" "^$" log,pass

SecFilterDefaultAction "deny,log,status:500"
```

SecAuditEngine On

SecAuditLog logs/audit_log

However, this will log all requests sent to the web server. Obviously, this can generate quite a lot of data very quickly. To only log requests that triggered a filter rule, set the `SecAuditEngine` variable to `RelevantOnly`. Alternatively, you can set this variable to `DynamicOrRelevant`, which will log requests to dynamic content or requests that triggered a filter rule.

As with most other Apache configuration directives, you can enclose *mod_security* configuration directives within a `<Location>` tag to specify individual configurations for specific scripts or directory hierarchies.

mod_security is a very powerful tool for protecting your web applications, but it should not take the place of actually validating input in your application or other secure coding practices. If at all possible, it is best to employ such methods in addition to using a tool such as *mod_security*.

See Also

- "Introducing mod_security": http://www.onlamp.com/pub/a/apache/2003/11/26/mod_security.html
- Mod_security Reference Manual v1.7.4: http://www.modsecurity.org/documentation/modsecurity-manual-1.7.4.pdf

```
### Windows computers

create windows-web

set windows-web personality "MS Windows2000 Professional RC1/W2K
Advance Server Beta3"

set windows-web default tcp action reset set windows-web default udp
action reset add windows-web tcp port 80 "perl scripts/win2k/iisemulator-
0.95

/iisemul8.pl"

add windows-web tcp port 139 open add windows-web tcp port 137 open
add windows-web tcp port 5900 "sh scripts/win2k/vnc.sh"

add windows-web udp port 137 open add windows-web udp port 135 open
create windows-xchng

set windows-xchng personality "MS Windows2000 Professional RC1/W2K
Advance Server Beta3"

set windows-xchng default tcp action reset set windows-xchng default udp
action reset add windows-xchng tcp port 25 "sh scripts/win2k/exchange-
smtp.sh"

add windows-xchng tcp port 110 "sh scripts/win2k/exchange-pop3.sh"

add windows-xchng tcp port 119 "sh scripts/win2k/exchange-nntp.sh"

add windows-xchng tcp port 143 "sh scripts/win2k/exchange-imap.sh"

add windows-xchng tcp port 5900 "sh scripts/win2k/vnc.sh"

add windows-xchng tcp port 139 open add windows-xchng tcp port 137
open add windows-xchng udp port 137 open add windows-xchng udp port
135 open ### Linux 2.4.x computer
```

```
create linux

set linux personality "Linux 2.4.7 (X86)"

set linux default tcp action reset set linux default udp action reset add linux
tcp port 110 "sh scripts/pop3.sh"

add linux tcp port 25 "sh scripts/smtp.sh"

add linux tcp port 21 "sh scripts/ftp.sh"

bind 192.168.0.10 windows-web bind 192.168.0.11 windows-xchng bind
192.168.0.12 linux

# <b>arpd 192.168.0.10-192.168.0.12</b> # <b>cd
/usr/local/share/honeyd</b> # <b>honeyd -p nmap.prints -x xprobe2.conf -
a nmap.assoc \</b> <b> -0 pf.os -f honeyd.conf</b> honeyd[5861]: started
with -p nmap.prints -x xprobe2.conf -a nmap.assoc -0 pf.os -f honeyd.conf

honeyd[5861]: listening on eth0: (arp or ip proto 47 or (ip )) and not ether
src 00:0c:29:e2:2b:c1

Honeyd starting as background process

# <b>nmap -sS -sU -O 192.168.0.10-12</b> Starting nmap V. 3.00 (
www.insecure.org/nmap/ ) Interesting ports on (192.168.0.10): (The 3063
ports scanned but not shown below are in state: closed) Port State Service

80/tcp open http

135/udp open loc-srv

137/tcp open netbios-ns

137/udp open netbios-ns

139/tcp open netbios-ssn

5900/tcp open vnc
```

```
Remote operating system guess: MS Windows2000 Professional RC1/W2K

   Advance Server Beta3

Uptime 2.698 days (since Sun Jan 11 03:52:35 2004) Interesting ports on
(192.168.0.11): (The 3060 ports scanned but not shown below are in state:
closed) Port State Service

25/tcp open smtp

110/tcp open pop-3

119/tcp open nntp

135/udp open loc-srv

137/tcp open netbios-ns

137/udp open netbios-ns

139/tcp open netbios-ssn

143/tcp open imap2

5900/tcp open vnc

Remote operating system guess: MS Windows2000 Professional RC1/W2K
Advance Server Beta3

Uptime 2.172 days (since Sun Jan 11 16:29:38 2004) Interesting ports on
(192.168.0.12): (The 1598 ports scanned but not shown below are in state:
closed) Port State Service

21/tcp open ftp

25/tcp open smtp

110/tcp open pop-3
```

```
Remote operating system guess: Linux 2.4.7 (X86)

$ <b>telnet 192.168.0.11 25</b> Trying 192.168.0.11...

Connected to 192.168.0.11.

Escape character is '^]'.

220 bps-pc9.local.mynet Microsoft ESMTP MAIL Service, Version:
5.0.2195.5329 ready at Mon Jan 12 12:55:04 MST 2004

EHLO kryten

250-bps-pc9.local.mynet Hello [kryten]

250-TURN

250-ATRN

250-SIZE

250-ETRN

250-PIPELINING

250-DSN

250-ENHANCEDSTATUSCODES

250-8bitmime

250-BINARYMIME

250-CHUNKING

250-VRFY

250-X-EXPS GSSAPI NTLM LOGIN

250-X-EXPS=LOGIN
```

```
250-AUTH GSSAPI NTLM LOGIN

250-AUTH=LOGIN

250-X-LINK2STATE

250-XEXCH50}
```

250 OK

```
add linux tcp port 22 proxy 192.168.0.100:22
```

In addition to running the service emulation scripts, *honeyd* can limit inbound or outbound bandwidth, or even slow down access to a particular service. This can be used to tie up spammer's resources, by holding open an apparently open mail relay. The possibilities provided by honeyd are limited only by your imagination and the time you're willing to spend building your virtual fly-catching network.

```
$ <b>./configure</b>

$ <b>make</b>

$ <b>tar tf sebek-linux-2.1.4-bin.tar </b>

sebek-linux-2.1.4-bin/


sebek-linux-2.1.4-bin/sebek.o

sebek-linux-2.1.4-bin/cleaner.o

sebek-linux-2.1.4-bin/sbk_install.sh

# <b>sh sbk_install.sh</b>

Installing Sebek:


   sebek.o installed successfully

   cleaner.o installed successfully

   cleaner.o removed successfully

$ <b>./configure && make</b>
```

After compilation has finished, become root and run `make install`. This will install *sbk_extract*, *sbk_ks_log.pl*, and *sbk_upload.pl*. To extract information sent from a honeypot, use *sbk_extract*. You can run it in sniffer mode by using the `-i` and `-p` options to specify which interface to listen on and which destination port to look for, respectively. If you want to process packets that have already been captured using a packet capture tool, use the `-f` option to specify the location of the packet dump file. Once you've extracted the data, you can use *sbk_ks_log.pl* to display the attacker's keystrokes.

Sebek also has an optional web interface that uses PHP and MySQL to allow more complex queries of the collected data. In addition to logged keystrokes, the web interface can extract files that have been uploaded to the honeypot. The *sbk_upload.pl* script uploads the logs to the web interface. Installation of the web interface is a bit more involved, since it requires an Apache server, PHP, and a MySQL 4 database. For more details, consult *Sebek*'s homepage at http://www.honeynet.org/tools/sebek/ .

Chapter 8. Recovery and Response

Hacks #96-100

Incident recovery and response is a very broad topic, and there are many opinions on the proper methods to use and actions to take once an intrusion has been discovered. Just as the debate rages on regarding vi versus emacs, Linux versus Windows, and BSD versus everything else, there is much debate in the computer forensics crowd on the "clean shutdown" versus

"pull the plug" argument. A whole book could be written on recovering from and responding to an incident since there are many things to consider when doing so, and the procedure you should use is far from well defined.

With this in mind, this chapter is not meant to be a guide on what to do when you first discover an incident, but it does show you how to perform tasks that you might decide to undertake in the event of a successful intrusion. In reading

this chapter, you will learn how to properly create a filesystem image to use for forensic investigation of an incident, methods for verifying that files on your system haven't been tampered with, and some ideas on how to quickly track down the owner of an IP address.

```
# <b>md5sum /dev/hda2 > /tmp/hda2.md5</b>

# <b>dd if=/dev/hda of=/tmp/hda.img</b>

# <b>fdisk -l -u /dev/hda</b>

Disk /dev/hda: 4294 MB, 4294967296 bytes

255 heads, 63 sectors/track, 522 cylinders, total 8388608 sectors

Units = sectors of 1 * 512 = 512 bytes


   Device Boot Start End Blocks Id System

/dev/hda1 * 63 208844 104391 83 Linux


/dev/hda2 208845 7341704 3566430 83 Linux

/dev/hda3 7341705 8385929 522112+ 82 Linux swap

# <b>dd if=hda.img of=hda2.img bs=512 skip=208845 count=$[7341704-
208845]</b>

7132859+0 records in


7132859+0 records out

# <b>md5sum hda2.img > /tmp/hda2.img.md5 && diff /tmp/hda2.md5
/tmp/hda2.img.md5</b>
```

The checksum for the image matches that of the actual partition exactly, so we know we have a good copy. Now you can rebuild the original machine and look through the contents of the copy at your leisure.

```
$ <b>make release</b>

$ <b>cd ..</b>



$ <b>cp ./install/install.cfg .</b>

$ <b>cp ./intall/install.sh</b>

# <b>sh ./install.sh</b>

# <b>twadmin --print-cfgfile</b> <b>> /etc/tripwire/twcfg.txt</b>

# <b>twadmin --create-cfgfile --site-keyfile</b> <b>./site.key
twcfg.txt</b>

# <b>tripwire --init</b>

# <b>twadmin --print-polfile > twpol.txt</b>

# <b>tripwire --update-policy twpol.txt</b>

# <b>tripwire --check</b>

# <b>twprint --print-report --twrfile \ </b>

<b>/var/lib/tripwire/report/colossus-20040102-205528.twr</b>

# <b>tripwire --update --twrfile</b> <b>\</b>

<b>/var/lib/tripwire/report/colossus-20040102-205528.twr</b>
```

You can and should schedule *Tripwire* to run its checks as regularly as possible. In addition to keeping your database in a safe place, such as on a CD-ROM, you'll also want to make backup copies of your configuration, policy, and keys. Otherwise you will not perform an integrity check in the event that someone (malicious or not) deletes them.

See Also

- `twpolicy (4)`
- The section "Using Tripwire" in Building Secure Servers with Linux, by Michael D. Bauer (O'Reilly)

```
<b>rpm -V</b> <span class="docEmphBoldItalic">package</span>
```

```
# <b>which ps</b> /bin/ps
```

```
# <b>rpm -V `rpm -qf /bin/ps`</b> S.5....T /bin/ps
```

```
# <b>rpm -Va</b> S.5....T /bin/ps
```

```
S.5....T c /etc/pam.d/system-auth S.5....T c /etc/security/access.conf S.5....T
c /etc/pam.d/login S.5....T c /etc/rc.d/rc.local S.5....T c
/etc/sysconfig/pcmcia .......T c /etc/libuser.conf S.5....T c /etc/ldap.conf
.......T c /etc/mail/sendmail.cf S.5....T c /etc/sysconfig/rhn/up2date-uuid
.......T c /etc/yp.conf
```

```
S.5....T /usr/bin/md5sum
```

```
.......T c /etc/krb5.conf
```

```
<b>rpm -qf</b> <span class="docEmphBoldItalic">filename</span>
```

```
<b>rpm -Vp </b> <span class="docEmphBoldItalic">package file</span>
```

RPM can be used for quite a number of useful things, including verifying the integrity of system binaries. However, it should not be relied on for this purpose. If at all possible, something like *Tripwire* [Hack #97] or AIDE (http://sourceforge.net/projects/aide) should be used instead.

```
# <b>./chrootkit</b> ROOTDIR is `/'

Checking `amd'... not found

Checking `basename'... not infected Checking `biff'... not found Checking
`chfn'... not infected Checking `chsh'... not infected Checking `cron'... not
infected Checking `date'... not infected Checking `du'... not infected
Checking `dirname'... not infected Checking `echo'... not infected Checking
`egrep'... not infected Checking `env'... not infected Checking `find'... not
infected Checking `fingerd'... not found Checking `gpm'... not infected
Checking `grep'... not infected Checking `hdparm'... not infected Checking
`su'... not infected
```

```
# <b>./chrootkit -r /mnt/hda2_image</b>
```

```
# <b>./chrootkit -p /mnt/cdrom</b>
```

You can also add multiple paths by separating each one with a :. Instead of maintaining a separate copy of each of these binaries, you could simply keep a statically compiled copy of BusyBox handy (http://www.busybox.net). Intended for embedded systems, BusyBox can perform the functions of over 200 common binaries, and does so using a very tiny binary with symlinks. A floppy, CD, or USB keychain (with the read-only switch enabled) with *chkrootkit* and a static BusyBox installed can be a quick and handy tool for checking the integrity of your system.

```
$ <b>whois badguydomain.com</b> Registrant:

   Dewey Cheatum Registered through: GoDaddy.com Domain Name:
BADGUYDOMAIN.COM

   Domain servers in listed order: PARK13.SECURESERVER.NET

   PARK14.SECURESERVER.NET

   For complete domain details go to: http://whois.godaddy.com

# <b>whois -h whois.arin.net 208.201.239.103</b> [Querying
whois.arin.net]

[whois.arin.net]

Final results obtained from whois.arin.net.

Results:

UUNET Technologies, Inc. UUNET1996B (NET-208-192-0-0-1)
208.192.0.0 - 208.255.255.255

SONIC.NET, INC. UU-208-201-224 (NET-208-201-224-0-1)
208.201.224.0 - 208.201.255.255

# ARIN WHOIS database, last updated 2004-01-18 19:15

# Enter ? for additional hints on searching ARIN's WHOIS database.

# <b>whois -h whois.arin.net NET-208-201-224-0-1</b> Checking server
[whois.arin.net]

Results:

OrgName: SONIC.NET, INC.

OrgID: SNIC
```

```
Address: 2260 Apollo Way

City: Santa Rosa

StateProv: CA

PostalCode: 95407

Country: US

ReferralServer: rwhois://whois.sonic.net:43

NetRange: 208.201.224.0 - 208.201.255.255

CIDR: 208.201.224.0/19

NetName: UU-208-201-224

NetHandle: NET-208-201-224-0-1

Parent: NET-208-192-0-0-1

NetType: Reallocated

Comment:

RegDate: 1996-09-12

Updated: 2002-08-23

OrgTechHandle: NETWO144-ARIN

OrgTechName: Network Operations OrgTechPhone: +1-707-522-1000

OrgTechEmail: noc@sonic.net # ARIN WHOIS database, last updated
2004-01-18 19:15

# Enter ? for additional hints on searching ARIN's WHOIS database.

alias whois='whois -h whois.geektools.com'
```

Now when I run `whois` from the command line, I don't need to remember the address of a single whois server. The folks at geektools have a bunch of other nifty tools to make sysadmin tasks easier. Check them out at http://geektools.com .

Rob Flickenger

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The image on the cover of *Network Security Hacks* is barbed wire. The type of barbed wire pictured in the cover image was patented by Joseph Glidden in 1874. Glidden improved on earlier attempts at manufacturing wire fencing by fashioning sharp barbs, spacing them along a smooth wire, and then twisting another wire around the first to hold the barbs in place. Advertised as "Cheaper than dirt and stronger than steel," barbed wire was immediately adopted by farmers in the American west as a way to control their herds. The days of free-roaming cattle and cowboys were soon numbered, but battles over barbs were fought both in court and on the ranch. Opponents called barbed wire "the Devil's rope," and the Cole Porter song "Don't Fence Me In" mourned this change in the western landscape. Barbed wire was here to stay, though--in addition to agricultural use, it has become a ubiquitous component of warfare and is a common feature of high-security areas such as prisons.

Genevieve d'Entremont was the production editor and copyeditor for *Network Security Hacks* . Brian Sawyer proofread the book. Philip Dangler and Claire Cloutier provided quality control. Jamie Peppard provided production support. Ellen Troutman-Zaig wrote the index. Rob Flickenger wrote the Preface.

Hanna Dyer designed the cover of this book, based on a series design by Edie Freedman. The cover image is a photograph from *gettyimages.com* . Emma Colby produced the cover layout with QuarkXPress 4.1 using Adobe's Helvetica Neue and ITC Garamond fonts.

Melanie Wang designed the interior layout, based on a series design by David Futato. This book was converted by Andrew Savikas to FrameMaker 5.5.6 with a format conversion tool created by Erik Ray, Jason McIntosh, Neil Walls, and Mike Sierra that uses Perl and XML technologies. The text font is Linotype Birka; the heading font is Adobe Helvetica Neue Condensed; and the code font is LucasFont's TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano and Jessamyn Read using Macromedia FreeHand 9 and Adobe Photoshop 6. This colophon was written by Philip Dangler.

The online edition of this book was created by the Safari production group (John Chodacki, Becki Maisch, and Ellie Cutler) using a set of Frame-to-XML conversion and cleanup tools written and maintained by Erik Ray, Benn Salter, John Chodacki, Ellie Cutler, and Jeff Liggett.

Avaya Labs, LibSafe technology

LZO compression

configuration program, using

number registries for IP address blocks

finding compromised packages

RRDtool