Sys Admin
the journal for UNIX systems administrators

Security Tools in FreeBSD

Guy Helmer

See Sidebar

Any UNIX system administrator or network manager on a tight budget, but with a need for stable, high-performance network servers, is prompted to consider alternatives to conventional commercial operating systems. While FreeBSD and similar systems obviously offer inexpensive alternatives for UNIX-like workstations and servers, their suitability from a security perspective may be less obvious. This article explores FreeBSD's security features with an eye toward assisting you in making a decision about whether it is suitable for your requirements.

Before launching into an exploration of FreeBSD, suffice it to say that the relationship of UNIX to Internet security is a massive topic. There are numerous books and other resources that discuss various elements of the subject, some of which are included in the Resource List at the end of this article. In particular, the interested reader is referred to *Practical UNIX and Internet Security* [5] and *Firewalls and Internet Security* [4], which are two of many well-known books on this topic.

Features of FreeBSD and Relatives

FreeBSD offers several features not usually found in stock UNIX-like systems. Some of these features are common to other BSD-derived systems (NetBSD and OpenBSD) or Linux, so one may wish to consult the system's documentation to see whether these features are available.

System Security Levels

UNIX system security has suffered from the all-powerful access given to the root-privileged user. Once a cracker obtains root privileges, the entire system becomes vulnerable to the cracker. The 4.4BSD-derived systems offer a new security feature called "system security levels" (see the man page for **init(8)** for details on these security levels). Raised system security levels prevent certain actions on a system irrespective of a user's privilege. Used properly, this can prevent an assortment of system compromises, including the introduction of Trojan horses and back doors into system binaries and modification of configuration files.

System security levels are: -1, permanently insecure mode; 0, insecure mode - no additional protections enabled; 1, secure mode - files may be protected from modification and memory device special files may not be opened for writing; 2, highly secure mode - protection of level 1 plus disk devices may not be opened for writing; 3, extended secure mode - protection of level 2 plus the IP packet filter list may not be changed. The default system security level of -1 is called "permanently insecure mode" because **init(8)** will not automatically raise the level to 1 when the system goes into multiuser mode as it would if the security level were 0.

The default system security level is -1, which corresponds to "normal" security for a traditional UNIX system. Raising the security level above 0 causes the kernel to disallow the following operations:

- Changes to any file whose immutable flag is set (see the **chflags(1)** man page for the **schg** flag).
- Removal of the immutable and append-only flags on files.
- Direct writes to disk devices which are mounted at security level 1, or direct writes to all disk devices at security levels higher than 1.
- Writes to the **/dev/mem** and **/dev/kmem** device special files.
- Loading any loadable kernel module.
- Changes to the IP packet filter lists (at security levels greater than 2).

When a FreeBSD system is installed by using a make world on the source code, a number of system files are installed with the immutable flag set. In general, executable files that are **setuid** root are installed with the immutable flag. Note that systems installed via **sysinstall** (e.g., installed via the network or from CD-ROM) do not have the immutable flag set on any files by the installation program. At any time, the **schg** flag may be set on files with **chflags**, but the flag can only be cleared when the system is at security level -1 or 0. During multi-user operation, the system security level may only be raised. The command to raise the security level to level 1 is:

```
sysctl -w kern.securelevel 1
```

The system security level can be raised automatically at system startup by including the above **sysctl** command in the system's **/etc/rc.local** file (and **/etc/rc.local** should then have its **schg** flag set as well). Raising the security level will limit file management activities, so you may not wish to use this feature on frequently changing systems. Once the security level has been raised, the only way to lower the security level is to restart the system. If one needs to perform maintenance on the system, it can be booted into single user mode where the security level will be its default of -1.

IP Firewall

Several TCP/IP protocols included with FreeBSD, such as NFS and NIS, may be more secure if they are unaccessible from outside the trusted part of the network. For protocols that may not be restricted any other way, a packet filter that allows only designated protocols can be used to block external access. The book *Building Internet Firewalls* [3] is suggested as a good introduction to IP packet filtering.

FreeBSD's kernel option IPFIREWALL enables IP firewalling, which applies filter rules to IP packets entering or exiting the machine. When a kernel built with the IPFIREWALL option starts, its default action is to drop every IP packet; additional filter rules must be given to a system to allow it to accept desired IP packets. The startup script **/etc/rc.firewall** provides sample filters, which must be modified before they can be used. The sample filter types are selected by the firewall option in **rc.conf**, which include:

open - No limitations on any protocols; any IP packet may enter or exit any interface.

client - Protects a system which is a client on a network. Given the client system's network number, subnet mask, and IP address, a basic filter is set up to allow all traffic originating from or destined for the local network, to allow packets for any established TCP connection, to allow incoming email and any outgoing TCP connection, and to allow Domain Name Service and Network Time Protocol UDP packets (everything else is denied).

simple - Set up a simple packet filter on a system that routes between networks, which is suitable for a simple FreeBSD router with two network interfaces. Given the network numbers, subnet masks, and IP addresses of the external ("unsafe") network and internal ("safe") network, this option in **rc.firewall** creates a packet filter that:

- Prevents outside packets with a source IP address of the internal network from entering, and likewise prevents packets from going outside unless they have a source address of the internal network (to prevent spoofing),
- Allows all packets for established TCP sessions,
- Allows TCP connections for incoming email, DNS, and World Wide Web requests,
- Rejects and logs all attempts to connect to TCP services other than those allowed above,
- Allows all DNS and Network Time Protocol UDP requests and responses,
- Denies anything not explicitly allowed.

If the system security level is above 2, changes are not allowed to the packet filter list. This allows a system administrator to prevent modifications to the packet filter list in the event of a system compromise, much like the raised security level can prevent modifications to parts of the filesystem. The IP firewall option can assist in monitoring potential attacks on a system. The kernel option IPFIREWALL VERBOSE enables logging of packet filtering messages via **syslogd(8)**. The logged information can be useful in spotting port scanning or break-in attempts.

One-Time Passwords

Password sniffing has been plaguing the Internet for some time [2]. Users logging into **telnet** or **ftp** across the Internet may be well advised to use one-time

passwords for logins from remote sites to avoid having their reusable password captured and abused by crackers. FreeBSD uses the S/Key software [6] to provide one-time passwords. Users can set up one-time passwords on a FreeBSD system by using the command:

```
keyinit
```

which will ask for a private password (which probably should be different than the regular system login password, but doesn't have to be) that only the user knows, and then **keyinit** will show something like:

```
ID fred s/key is 99 sp99609
BUY OUR BOLD LEER YOKE COW
```

Then, the user can generate a series of one-time passwords to take on her travels by using the command:

```
key -n 10 98 sp99609
```

which will ask for the user's private password used in the **keyinit** step above and generate the one-time passwords for the next 10 logins (98 down to 89). The user can print these passwords and take them along for her next remote logins over the Internet. (Of course, the printed password list poses other security concerns.)

A FreeBSD system can require the use of S/Key passwords for logins from all sites other than local systems. The file **/etc/skey.access** (see the **skey.access(5)** manual page for complete information about this file) defines where S/Key passwords must be used. For example, if the administrator trusts the systems in his own network (numbered 172.16), but wants users to always use S/Key one-time passwords for logins from other sites, he could use a **/etc/skey.access** file like this:

```
permit 172.16.0.0 255.255.0.0
deny
```

Noe that **/etc/skey.access** does not affect all remote access methods, just **telnet**, **ftp**, and **rlogin**. For example, **ssh**, **pop** daemons, and **imap** daemons ignore S/Key restrictions.

Disallowing Logins

An administrator can completely restrict **telnet** and **ftp** access for particular users and sites by adjusting the **/etc/login.access** file. This option can reduce the usefulness of compromised passwords by limiting the sites from which a system can be abused (see the **login.access(5)** manual page for details on this file).

Note that **login.access** does not affect all remote access methods, just **telnet**, **ftp**, and **rlogin**. For example, **ssh**, **pop** daemons, and **imap** daemons ignore the restrictions in **login.access**.

For example, if an administrator wanted to allow logins for all accounts to his or her system from two networks, 172.16 and 192.168.32, and allow all users in the group wheel to also login from the system with IP address 192.168.2.2, this **/etc/login.access** file could be used:

```
+:ALL:172.16.
+:ALL:192.168.32.
+:wheel:192.168.2.2
-:ALL:ALL
```

Conclusion

Thanks to its development team and support from related developers, FreeBSD is a solid system that can be configured to stand up to security attacks. While the default installation is still fairly open and friendly to users, FreeBSD provides many security mechanisms and several add-on packages that can improve a system's resistance. These security improvements can ensure the operation and availability of the system as well as the confidentiality of the information it contains.

Acknowledgments

Thanks to Adam Shostack, Falko Dressler, Robert N. Watson, and other anonymous reviewers for significant comments on previous versions of an article on which this article was based. Thanks to the FreeBSD core team and all of the FreeBSD contributors who have made FreeBSD an excellent performing, extremely stable operating system. Thanks also to the developers who produce security fixes, usually within days after security issues are found.

References

[1] Steven M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19(2), April 1989.

[2] CERT Advisory 94.01: *Ongoing network monitoring attacks*. [Online] **ftp://info.cert.org/pub/cert- **
**advisories/CA-94:01.network. **
monitoring.attacks, August 17 1997.

[3] Brent Chapman and Elizabeth Zwicky. 1995. *Building Internet Firewalls*. O'Reilly & Associates, Inc., Sebastopol, CA.

[4] Bill Cheswick and Steve Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. 1994. Addison-Wesley.

[5] Simson Garfinkel and Eugene H. Spafford. 1996. *Practical UNIX and Internet Security*, Second Edition. O'Reilly & Associates, Inc., Sebastopol, CA.

[6] Neil M. Haller. The S/KEY one-time password system. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, San Diego, CA. February, 1994.

[7] Robert T. Morris. A weakness in the 4.2BSD UNIX TCP/IP software. *Science of Computer Programming*, February 25, 1985.

[8] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*, February, 1988. (Version 4).

[9] Wietse Venema. TCP wrapper: Network monitoring, access control, and booby traps. In *Proceedings of the UNIX Security III Symposium*. USENIX, September, 1992.

About the Author

Guy Helmer is a graduate student in Computer Science at Iowa State University, concentrating on security in operating systems and networks. Previously, Guy was a system programmer and network engineer for a university, where he engineered local and wide area networks and administered UNIX servers, including several FreeBSD servers. Guy has been a beta tester and occasional contributor for the FreeBSD Project since its inception.