Staying one step ahead of the intruders

PREVENTION

Internet intruders have many ingenious ways of escalating privileges and hiding their presence once they get inside your system. The best protection is to keep them out in the cold.

BY TIM SCHÜRMANN AND JOE CASAD

Just when you think you've mastered the art of intrusion protection, the cyber-criminals discover new some techniques for slipping through your security. Attackers use every possible advantage to stay hidden and gain control. Shouldn't you use every trick to keep them out?

This month's cover story focuses on techniques for keeping intruders off your system. In our lead-off article, "Who's There: Single Packet Port Knocking with fwknop," we study a powerful technique that lets you keep your firewall ports closed to all unauthorized users – but still open to traffic from friends. The next article, "Closing the Book: Fighting Dictionary Attacks with Sshutout and Fail2ban," discusses a pair of tools that will help you keep intruders from guessing your passwords.

Next, we show you how to apply more flexible and precise permissions to files and other objects using Access Control Lists (ACLs). We end with a workshop on the powerful security tool known as SELinux.

Getting In

Read on for more about locking down your Linux system – but first we take a quick look at exactly why this elaborate dance of security is even necessary. The rest of this introduction examines rootkits in the Linux environment.

What if someone broke into your Linux system and replaced the *login* program with a malignant variant. The new *login* finds out your username and password and sends the data it collects through a hole it has punched in your firewall to a server somewhere in the wild. No one suspects the new *login*, although the intruder suspects that an attentive system administrator might eventually wonder about the change in the file size.

But Internet criminals have a way of covering their tracks. Along with the manipulated *login* tool, the attacker smuggles in a modified *ls* program. This new version of *ls* masks any changes to the size or modification date for the *login* program.

The attacker also decides to replace several other system programs that work together to gather information and hide any evidence of the intrusion. Even antivirus programs are powerless because the manipulated tools fool them, too.

This scenario is by no means fiction. Attackers often bring along a collection of tools that captures information, opens backdoors, and conceals their activities. This bundle of weapons is commonly known as a rootkit.

A rootkit

COVER STORY

Single-Packet Port Knocking 24
Sshutout and Fail2ban 30
Access Control Lists 32
SELinux 36

typically comprises multiple components that fill various needs:

- A trojan drops the rootkit and anchors it on the system.
- A sniffer analyzes the network traffic and retrieves the access credentials.
- In some cases, keyloggers log keyboard input to capture passwords or PINs before the system can encrypt them.
- A backdoor gives the attacker access to the system.

All of these activities are camouflaged by replacing system files and, for current rootkits, redirecting API calls. Other components then put the computer to use – possibly for distributing spam or performing denial of service attacks.

Innovation

In the early days, rootkits simply replaced popular system tools such as *ls*, *passwd*, or *ps*. Security experts soon learned how to detect these early rootkits, and malware programmers soon learned to target the kernel itself.

If an attacker manages to inject malicious code into the kernel, the offending kernel code can capture and redirect any request.

Rootkits running in kernel space are particularly hard to discover. On Linux, kernel rootkits are often injected through a kernel module, which explains why they are known as LKM (for Loadable Kernel Module) rootkits.

Rootkit developers use various approaches to infest the kernel. One option is to manipulate the memory directory via */dev/kmem*.

Firmware

Firmware rootkits provide an alternative attack vector. They infect the PC's firmware and survive a reboot. Some rootkits feel quite at home in the ACPI firmware routines. A clean rescue disk is little help against this kind of threat.

To Build, or Not to Build

Administrators are asked to build most rootkit scanners before they can use them. As you can imagine, this is a problem if the rootkit already has a stranglehold on your system. In this case, the compiler may already have been compromised by the rootkit, and would build a manipulated version of the scanner. For this reason, you should try to build the scanner as soon as possible after installing the system, or use a system that you know is clean.

Blue Pill Detection

The latest trend is virtualized rootkits. A virtualized rootkit works like a virtual machine: The first step is for the rootkit to modify the boot process so that a virtual machine is loaded before the operating system. The operating system you think is running on the hardware actually boots in a virtual machine. This gives the rootkit total control over the computer without the operating system or the user noticing a thing. This approach was demonstrated by Blue Pill [1], which explains why the process of identifying virtual rootkits is known as Blue Pill Detection.

Prevention

Rootkits are very difficult to detect once they are installed on your system. Before they install, however, you can identify them with anti-virus programs and rootkit scanners (or rootkit detectors), which use signatures or heuristics to identify the culprits.

The best way to stop a rootkit is not to let it in. The articles in this issue discuss some techniques for shutting out intruders before they get comfortable.

In spite of all your efforts, however, you will never be safe enough to ignore the possibility of a rootkit slipping past your defenses. The following sections discuss some strategies for rooting out rootkits.

Visual Check

Because the results on a running system are not entirely conclusive, you should down the suspect and then boot from a medium that you know is clean. This could be a rescue disk, for example. The real test is to cross check any suspicious files with files from a clean system. This is usually accomplished by comparing the current system with a snapshot of the system taken directly after installation. To prevent the rootkit from affecting the results of the comparison, store the original snapshot on a read-only medium.

Of course, a backup of a complete system takes significant disk space. As an alternative, you could just create checksums for an integrity check. To do this, the rootkit scanner calculates a fingerprint for each file; if the rootkit later attempts to manipulate the file, the checksum is automatically changed and will thus not match the original checksum stored at the time of the last upgrade.

Chkrootkit

One of the most popular rootkit scanners on Linux is Chkrootkit [2]. The toolkit by Nelson Murilo and Klaus Steding-Jessen comprises a collection of small C programs specially written to detect a specific anomaly. After unpacking the archive, build the applications by typing

```
make sense
```

and then perform a trial run by typing *./chkrootkit* (Figure 1). Unfortunately, the rootkit scanner calls various binary programs on the infected system. For this reason, you should always run these

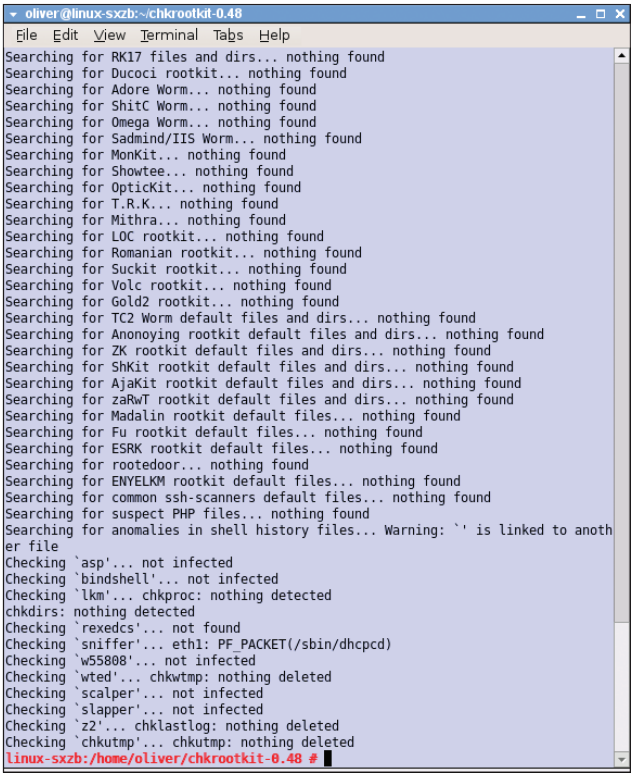

Figure 1: Chkrootkit has run once and not found a known rootkit.

tools directly from a separate, clean medium such as a CD or DVD drive:

```
./chkrootkit -p /cdrom/bin
```

in order to avoid infection.

Rootkit Hunter

Just like Chkrootkit, Rootkit Hunter also searches the infected system for specific characteristics that indicate the existence of a rootkit. Originally written by Michael Boelen, the tool was passed on to a team of developers in 2006, and the results are visible on SourceForge [3]. After downloading and unpacking the archive, become root and enter the following to build:

```
./installer.sh ➥
--layout custom . --install
```

Then change to the *files* subdirectory and modify the *rkhunter.conf* configuration file. The following command line starts the check

```
./rkhunter --check
```

OSSEC

In contrast to previous, simple scanners, OSSEC [4] launches a whole battery of functions (Figure 2). In addition to automatically performing period rootkit detection, it offers permanent monitoring and analysis of logfiles, integrity checks, and (rules-based) intrusion detection. The Rootcheck project has rootkit signatures up for grabs on its website [5].

OSSEC lets you set up a client/server team: Agents monitor the operating system on the client machines. Whenever a suspicious or atypical event occurs, a message is passed on to a central monitoring server, which then performs the analysis, draws conclusions, and raises the alarm if necessary.

To install OSSEC, just unpack the archive and type:

```
./install.sh
```

Say yes to all the prompts and choose *local* as the installation type, if you do not want to set up an agent/server operation.

Next, select one of the installation routine options for rootkit detection, then modify the configuration in */var/ossec/etc/ossec.conf*, and launch OSSEC HIDS by typing

```
/var/ossec/bin/➥
ossec-control start
```

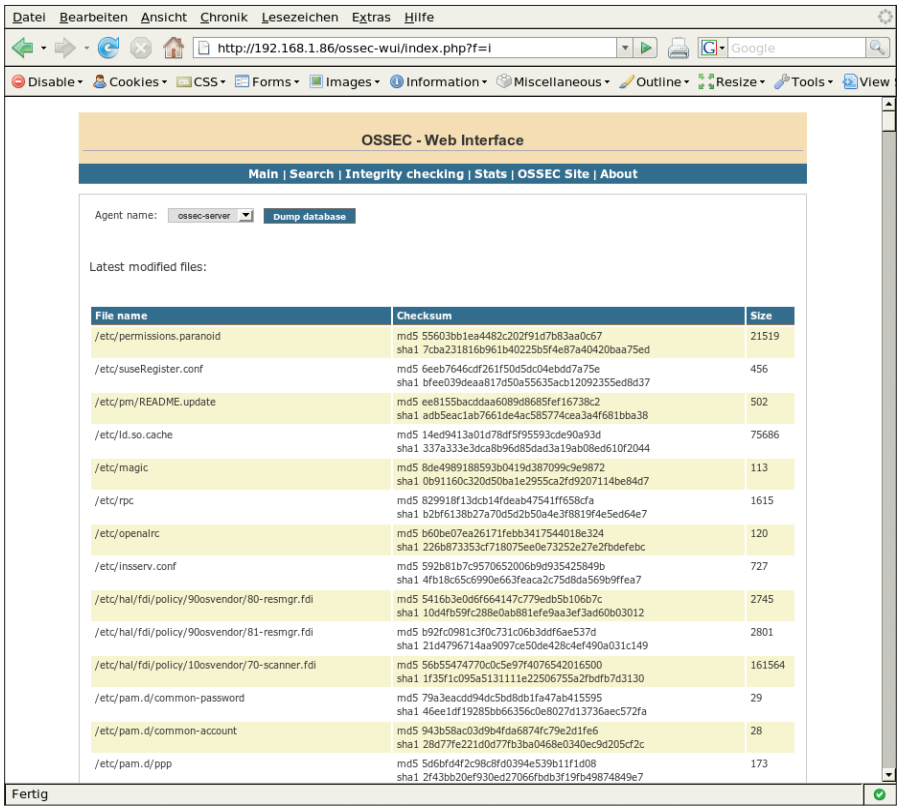

Figure 2: The OSSEC project offers a web front-end for the monitoring agent.

The program then starts to monitor the target system.

In a default installation, the configuration file is stored in */etc/ossec-init.conf*, with all other files in */var/ossec*. Rootkit signatures are stored in the files *rootkit_files.txt* and *rootkit_trojans.txt* in */var/ossec/etc/shared*.

Delousing

After detecting a rootkit, the only thing that really helps is to completely reinstall the infected system — the malicious software digs far too deep into a system to let you remove it with any degree of certainty. In this light, it makes sense to do everything you can to prevent a rootkit from compromising your system. Prevention is the only real answer – particularly, installing security patches. To prevent malware exploiting them, you need to make sure that you close all known vulnerabilities.

Latest and Greatest

The principle of using the latest software also applies to rootkit scanners. Zeppoo [7] is almost two years old, for example, and fails to detect any of the current crop of rootkits due to total ignorance of their approaches. It is probably best to leave Zeppoo where it is on SourceForge.

The next best obstacle is a firewall, preferably one that controls network traffic in all layers of the TCP/IP stack. For servers in hostile environments, the use of a static kernel and an intrusion detection tool such as OSSEC.

Keep in mind, however, that humans are possibly the biggest attack vector for rootkits. If a user is enticed by phishing mails or enlargement offers, the best security mechanisms are no doubt doomed to failure. ■

INFO

[1] Blue Pill: http://en.wikipedia.org/wiki/Blue_Pill%28malware%29

[2] Chkrootkit: http://www.chkrootkit.org

[3] Rootkit Hunter: http://rkhunter.sf.net

[4] OSSEC: http://www.ossec.net

[5] Rootcheck signatures: http://www.ossec.net/rootkits

[6] RK Profiler LX: http://www.trapkit.de/research/rkprofiler/index.html

[7] Zeppoo: http://sourceforge.net/projects/zeppoo