Invisible Intruders: Rootkits in Practice

DAVID BRUMLEY

Davd Brumley then...

David Brumley works for the Stanford University Network Security Team (SUNSeT). He graduated with honors from the University of Northern Colorado in mathematics with additional work in philosophy. In his free time he enjoys playing hockey and reading Kant.

...and now

David Brumley is an Associate Professor at Carnegie Mellon University in the Electrical and Computer Engineering Department. Prof. Brumley graduated from Carnegie Mellon University with a PhD in Computer Science in 2008. Hehas received several best paper awards, an NSF CAREER award, and the United Stated Presidential Early Career Award for Scientists and Engineers. dbrumley@cmu.edu

Reprinted from *;login:* Special Issue on Intrusion Detection, September 1999

To catch a cracker you must understand the tools and techniques he will use to try to defeat you. A system cracker's first goal is to hide his presence from you, the administrator. One of the most widely used cracker tools for doing this is the rootkit. A rootkit gets its name not because the toolbox is composed of tools to crack root, but instead because it comprises tools to keep root.

Rootkits are used by intruders to hide and secure their presence on your system. An intruder achieves complete cloaking capability by relying on an administrator to trust the output of various system programs. This assumption is more or less true – most of the time system administrators trust ps to display all processes and ls to list all files.

The cracker hides simply by modifying these programs not to display his activities: ls is altered not to display the cracker's files, and ps is modified not to display the cracker's processes. This simple method proves powerfully effective. A system administrator often has no clue that anything is amiss. Should the administrator sense that the system does not "feel" right, she'll have a hard time tracking down the exact problem.

To replace any of the programs mentioned here, the cracker must already have root access. The initial attack that leads to superuser access is often very noisy. Almost every exploit will produce a lot of network traffic and/or log activity. Once in, though, the skilled attacker has no difficulty covering tracks. The average cracker will have programs in his rootkit such as z2 and wted that remove login entries from the wtmp, utmp, and lastlog files. Other shell scripts may clean up log entries in /var/log and /var/adm. Luckily, the average cracker is sloppy. Sometimes he will forget to clean out certain programs or will simply just zero out the log file. Any time a log file has zero length it should be an immediate sign that something is amiss.

Trojans

Once the cracker cleans up the appropriate files to hide his tracks, he will want to leave a backdoor in order to avoid using his noisy exploit again. Rootkit backdoors – often called trojan horses – can typically be divided into two categories: local programs and network services. These trojaned programs are the core of the rootkit.

Local programs that are trojaned often include chfn, chsh, login, and passwd. In each case, if the magic rootkit password is entered in the appropriate place, a root shell is spawned. Of course a smart cracker will also disable the history mechanism in the root shell.

The replacement for login is especially interesting. Since some systems have shadowed and unshadowed password schemes, the cracker's replacement must be of the right type. A careless cracker might use the wrong kind of login trojan. When this happens, all or some accounts will be inaccessible, which should be an immediate tipoff that a cracker has gained control of your system.

inetd, the network super daemon, is also often trojaned. The daemon will listen on an unusual port (rfe, port 5002 by default in Rootkit IV for Linux). If the correct password is given after connection, a root shell is spawned and bound to the port. The manner in which the shell is bound makes it essential to end all commands with a semi-colon (";") in order to execute any command line.

rshd is similarly trojaned. A root shell is spawned when the rootkit password is given as the username (i.e., `rsh [hostname] -l [rootkit password]` will get you in to the compromised machine).

Last, a root shell is often simply left bound to a port by the program bindshell. This program requires no password. By default the program is bound to port 31337, "eleet" in cracker jargon.

Satori

In all of these programs, the default password for the newest Linux rootkit (Rootkit IV) is `satori`. Older rootkits have used `lrkr0x` and `h0tb0x` as passwords. Rarely is the default left unchanged, but it never hurts to check.

To expand their domain, the cracker may also install an Ethernet sniffer. An Ethernet sniffer listens in on all traffic on the local network, grabbing passwords and usernames destined for other machines. ifconfig will normally report such odd behavior by alerting the administrator with the PROMISC flag. Unfortunately, ifconfig is usually one of the programs modified.

The allure of rootkits should now be obvious. Even if the administrator patches the program that initially led to root access, the cracker merely has to telnet to the proper port to get a root shell. If this is closed, the cracker can try the backdoored login or rshd program. And even if that doesn't work, the cracker can still log in as a user (from perhaps a cracked password or his Ethernet sniffer) and used the trojaned ping, chfn, or chsh program to become the superuser once again.

Why do crackers break into systems? Sometimes you are targeted directly. The cracker wants information or access specifically available at your installation. Often, however, a cracker may simply want to break into any system in order to get on IRC, serve up WAREZ, or trade MP3s. If they do this, they might trojan crontab in order to hide jobs that rotate, modify, or check on the status of the illicit activity.

Hidden

What tools does the administrator have to find these trojan-horse programs? If a rootkit is properly installed, the administrator will not be able to tell the difference between the original and a modified program. A widely used cracker program named fix will take a snapshot of the system binary to be replaced. When the trojaned or modified binary is moved into place, the fix program mimics all three timestamps (atime, ctime, and mtime) and CRC checksum of the original program. A carefully constructed and compiled binary will also have the same length.

Without a cryptographically secure signature of every system binary, an administrator cannot trust that she has found the entire rootkit. If even one program goes undetected, the intruder might have a way back into your system. Several utilities, such as tripwire and RedHat's rpm, provide secure MD5 checksums of binaries. To be truly secure, the reports must be kept offline in some sort of secure location, lest the hacker tamper with the report. (Not so long ago a system-cracker magazine called Phrack published an article on defeating online tripwire reports.) These reports may be the only thing that saves you from a complete reinstallation of the entire system.

Luckily, many crackers are careless, and portions of their rootkit can be detected. The trojaned files above often have configuration files that list the programs to hide and which to display. Often they forget to hide the configuration files themselves. Since /dev is the default location for many of these configuration files, looking in there for anything that is not a normal file is often a good idea. The default setup for many rootkits is to have the configuration file begin with `pty`, such as `/dev/ptys` or `/dev/pryr`.

Another trick is to look at modification times of all programs. Although a good cracker will try to cover most of the times, they often forget a few files or directories. `find / -mtime -N -print`, where N is the number of days you expect the intruder has had access to your system, should work in most cases. I've found many times the hacker has covered his tracks well in /bin and /sbin, but left the entire build directory for his rootkit in /tmp!

Inside each modified directory you should compare the output of `echo *` with `ls`. If `ls` has been trojaned and configured to hide anything, the `echo` command will show it.

Also pay close attention to the strings in the system binaries. Although /sbin/inetd may look the right size, if the string "/bin/bash" shows up in it, you should start worrying about what else has been replaced. Another trick is to look at the file type. If file /bin/inetd says that inetd is not stripped, it most certainly has been tampered with.

If you're lucky enough to have a /proc filesystem, spend some time to become acquainted with it – there is a lot of useful

information there. By walking the directory tree you can find which processes are running. After comparing the output to what ps shows, you can determine with some level of certainty whether ps has been modified. Other files in /proc may show you all active network connections, and some others may even list all open file descriptors!

The easiest way to detect crackers, however, is to have a clean set of statically linked binaries for your system. Statically linked? Sometimes a more advanced cracker will replace system libraries, so anything that dynamically uses them cannot be trusted. If possible you should have a spare set of common programs such as ps, ls, ifconfig, perhaps lsof, etc., on a secure host. When you find a compromised system, simply download the clean binaries, set your PATH environment variable to use them, and start looking for backdoors.

Various versions of rootkit are available at most cracker sites. The most accessible versions are for open-source operating systems such as Linux and FreeBSD. Also commonly reported are versions for Irix, SunOS, and Solaris. The latest rootkit, Linux Rootkit IV, is distributed by The Crackers Layer, http://www.lordsomer.com. It is definitely worth the bandwidth to download the source and see how it works.

Rootkits have become very popular tools for both experienced and novice crackers. Your first line of defense should always be protection with regular patches and administration. Equally important is the second line: a good plan in the event of a real compromise. By arming yourself ahead of time with secure checksums and clean binaries, you will be much quicker and more effective in local and sitewide incident response.

Utilities Included in Rootkit IV

Programs That Hide the Cracker's Presence

ls, find, du—will not display or count the cracker's files.

ps, top, pidof—will not display the cracker's processes.

netstat—will not display the attacker's traffic, usually used to hide daemons such as eggdrop, bindshell, or bnc.

killall—will not kill the attacker's processes.

ifconfig—will not display the PROMISC flag when sniffer is running.

crontab—will hide the cracker's crontab entry. The hidden crontab entry is in /dev by default.

tcpd—will not log connections listed in the configuration file.

syslogd - similar to tcpd.

Trojaned Programs That Have Backdoors

chfn—root shell if rootkit password is entered in as new full name.

chsh—root shell if rootkit password is entered as new shell.

passwd—root shell if rootkit password is entered as current password.

login—will allow the cracker to log in under any username with the rootkit password. If root logins are refused, user rewt will work. It also disables history logging.

Trojaned Network Daemons

inetd—root shell listening on port rfe (5002). After connection, the rootkit password must be entered in as the first line.

rshd—trojaned so that if the username is the rootkit password, a root shell is bound to the port (i.e. rsh [hostname] -l [rootkit password]).

Cracker Utilities

fix—installs a trojaned program (e.g., ls) with the same timestamp and checksum information.

linsniffer—a network sniffer for Linux.

sniffchk—checks to make sure a sniffer is still running.

wted—wtmp editor. You can modify the wtmp.

z2—erases entries from wtmp/utmp/lastlog.

bindshell—binds a root shell to a port (port 31337 by default).