# Global Information Assurance Certification Paper

**Adding Chkrootkit to Your Unix Auditing Arsenal**

Bill Hutchison

February 26, 2001

## Introduction

When auditing any Unix system, it is advantageous to have a diverse selection of tools to help monitor a system for unusual and unexpected changes. As stated in the GIAC Level One Security Essentials Course, basic system auditing is the technique that is used to detect and record such changes in order to assure a system is secure. Those changes can be caused by any number of sources. The reports that are collected by regular audits assist in watching for intrusion attempts and discovering security break-ins. This same audit trail is also vital when assessing the damage that has been done after a system's security has been breached.

All Unix systems come equipped with a standard set of system utilities that can form the basis for any good auditing kit. There are also many credible free tools available which can assist with watching and reporting security-related changes. When looking at which tools to use during your system audits, it is important to use tools that are simple to implement and that complement the standard system utilities. It is equally important to have a tool that is multifaceted and one that can help with both regular security audits and with collecting forensic information when dealing with a break-in.

## Auditing systems for rootkit installations

One of the common tools in use by intruders after breaking into a system are rootkits. Once an intruder gains access to a system, they will use a popular rootkit to quickly install other hacking tools and to hide evidence of their initial break-in and occupancy. Rootkits are the means by which intruders increase their chances of keeping root access to a system.

Rootkits are created to conceal and mislead many of the tools used in a basic Unix audit. A rootkit will typically replace many of the common system utilities with hacked versions. These versions will report false data so to help to hide directories and running processes. They can even embed themselves directly into the Unix kernel for an even deeper level to evasion from more sophisticated security auditing tools.

This paper will specifically look at chkrootkit, and its use in detecting the presence of rootkits during security audits.

## What is Chkrootkit

Chkrootkit is a free software package that works with Linux, Solaris, FreeBSD, and OpenBSD. It is a lightweight utility with a very small installation footprint. Chkrootkit uses the knowledge of the security community's analysis of popular rootkits to detect

their fingerprints after used against a system. Chkrootkit is available from http://www.chkrootkit.org.

Chkrootkit detects the following known rootkits and worms as of version 0.22:

- Solaris rootkit
- FreeBSD rootkit
- lrk3, lrk4, lrk5, and lrk variants
- T0rn rootkit
- Ambient's Rootkit for Linux (ARK)
- Ramen Worm

Some of these attacker tools have been around for a few years, others, like the Ramen worm are as recent as January 2001. Chkrootkit has been in development since 1997.

The core component of chkrootkit is a shell script that orchestrates it's operation. It handles checking important system binaries against known rootkit signatures; a method similar to the one used by popular virus detection programs use for spotting infections.

These system files are checked for modification:

- chfn, chsh, cron, sshd, du, find, fingerd, su, ifconfig, inetd, killall, login, ls, netstat, passwd, pidof, ps, rshd, syslogd, tcpd, top, telnetd, sap, bindshell

Loadable Kernel Modules are also checked for trojans. This is a unique and interesting aspect of chkrootkit. A LKM is a facility made available by modern operating systems for easily extending the kernels abilities dynamically. It is also a very powerful tool for intruders to use to evade detection. To spot rogue LKMs that are hidding processes, Chkrootkit uses it's chkproc utility to see if entries in the /proc filesystem are hidden from ps and the readdir system call.

Chkrootkit is a very self-contained tool, a vital trait of any tool used for security auditing. The only external commands that chkrootkit relies on for successful operation are:

- awk, cut, egrep, find, head, id, ls, netstat, ps, strings, sed, uname.

**Command-line options and usage**

Executing chkrootkit with no options will cause it to execute all it's tests using utilities currently installed on the system. In addition, it will output it's results. Chkrootkit's command-line parameters are there to help either limit what tests it runs, control where chkrootkit gets it's support utilities, or whether it is run in an expert mode.

```
Usage: chkrootkit [options] [sub-test ...]
```

Options:

| | |
|---|---|
| -h, V, l | Display the help, version, and available tests and exit |
| -d | Put chkrootkit into debug/test mode |
| -x | Expert mode |
| -r dir | Specify the root directory for tests. Useful when inspecting a remote system via a NFS mount |
| -p dir1:dir2:dirN | The path(s) for the external commands used by chkrootkit |

An example command might be:

```
chkrootkit -p /cdrom/bin:/floppy/mybin
```

These parameters will cause chkrootkit to get the system commands that it relies on from /cdrom/bin and /floppy/mybin, rather than from their standard locations on the system being examined.

Or an example from chkrootkit's online FAQ:

```
chkrootkit -x | egrep '^/'
```

These parameters will run chkrootkit in expert mode and allow for locating pathname strings in system commands. The expert mode helps when looking for suspicious strings in binaries. Since chkrootkit's set of know signatures is fixed, this parameter helps to extend chkrootkit's abilities. Chkrootkit will output all matches for the user to examine for suspicious strings that may indicate a trojan.

Example of output from chkrootkit when run on a RedHat Linux 6.2 system:

```
#./chkrootkit
ROOTDIR is `/'
Checking `chfn'... Not vulnerable
Checking `chsh'... Not vulnerable
Checking `cron'... Not vulnerable
Checking `sshd'... Not vulnerable
Checking `du'... Not vulnerable
Checking `find'... Not vulnerable
Checking `fingerd'... Not vulnerable
Checking `su'... Not vulnerable
Checking `ifconfig'... Not vulnerable
Checking `inetd'... Not vulnerable
Checking `killall'... Not vulnerable
Checking `login'... Not vulnerable
Checking `ls'... Not vulnerable
Checking `netstat'... Not vulnerable
Checking `passwd'... Not vulnerable
Checking `pidof'... Not vulnerable
Checking `ps'... Not vulnerable
Checking `rshd'... Not vulnerable
Checking `syslogd'... Not vulnerable
```

```
Checking `tcpd'... Not vulnerable
Checking `top'... Not vulnerable
Checking `telnetd'... Not vulnerable
Checking `asp'... Not vulnerable
Checking `bindshell'... Not vulnerable
Checking `z2'...
Nothing deleted
Checking `wted'... Nothing deleted
Checking `sniffer'...
eth0 is not promisc
vmnet1 is not promisc
Checking `aliens'... No suspect files
Searching for sniffer's logs, it may take a while... Nothing found
Searching for t0rn's default files and dirs... Nothing found
Searching for Ambient's rootkit (ark) default files and dirs... Nothing
found
Searching for suspicious files and dirs, it may take a while...
/usr/lib/linuxconf/install/gnome/.directory
/usr/lib/linuxconf/install/gnome/.order /usr/lib/perl5/5.00503/i386-
linux/.packlist
/usr/lib/perl5/site_perl/5.005/i386-linux/auto/MD5/.packlist
/lib/modules/2.2.14-5.0/.rhkmvtag
Searching for Ramen Worm files and dirs... Nothing found
Checking `lkm'... Nothing detected
```

Looking at the output above, chkrootkit first checks all the system commands It will also search for sniffer logs, hidden directories, and rootkit config files. It also controls a few provided utilities (chkwtmp, chklastlog, ifpromisc) that test wtmp and lastlog for deletions and to check if your network interface is running in promiscuous-mode.

**Chkrootkit as an auditing tool**

Chkrootkit obviously cannot serve as the only auditing tool used on your Unix systems. But it should be one of the first ones run during your regular audit process. Running it early in a system audit allows you to check your basic system commands and to look for rogue LKMs from a known good source quickly. You can then run the rest of your auditing tasks that use basic Unix tools like ps, netstat, find, etc., with the extra knowledge that no known rootkit exploit is concealing their output.

Chkrootkit also continues to expand it's set of rootkit detection signatures. Using chkrootkit on a regular basis will insure your systems a growing list of rootkits to test against. Like most viruses, specific rootkits will be in use for quite some time and you must continually check for their presence, no matter how old they may be.

Chkrootkit's small footprint allows for installing its components onto a single floppy disk. Therefore, you can easily automate the use of chkrootkit by putting it on a self-contained locked floppy disk and setting up a nightly cron job that will e-mail its results.

**Chkrootkit as a forensic tool**

When examining a system break-in, it's handy to quickly narrow down what an intruder has done. Chkrootkit provides a convenient means for discovering either exactly what damage has been done, or the opposite, helping you to throw out a known set of tools that were not used. This helps to establish a baseline for where to look in your investigation.

Since the system under investigation cannot be relied apon as a source for running executables, using the -p parameter and specifying a floppy disk as the source for all support utilities, chkrootkit can be quickly run against a system that is suspected of a break-in.

**Conclusion**

Protecting your systems from rootkits requires more than a single tool. It requires planning upfront when installing new systems, maintaining good backups, keeping up-to-date on system patches, and performing regular audits. Chkrootkit is a tool that can provide another viewpoint during those audits.

It is generally up to the system administrator to collect the set of tools to be used during an audit. But, given this added workload, some may tend to pass up basic host auditing in favor of popular all-in-one security products such as firewalls and intrusion detection systems. But when these systems fail, the next line of defense is a system's audit trail.

Keeping auditing tasks simple and straightforward helps to make sure they will get done on a regular basis. A tool, such as chkrootkit, that provides a good and quick examination of a system with very little overhead would make a useful addition to any basic Unix auditing kit.

**Sources**

Chkrootkit website. http://www.chkrootkit.org

Murilo, Nelson. Chkrootkit source files. Version 0.22. 25 Jan 2001.

pragmatic / THC, "(nearly) Complete Linux Loadable Kernel Modules ", Jan 1999. http://packetstorm.securify.com/docs/hack/LKM_HACKING.html

"Welcome to the Linux Rootkits section." http://packetstorm.securify.com/UNIX/penetration/rootkits/

Houle, Kevin. "CERT® Incident Note IN-2001-01." Widespread Compromises via ramen Toolkit18 Jan 2001. http://www.cert.org/incident_notes/IN-2001-01.html

Brumley, David. "Rootkits - How Intruders Hide." http://www.theorygroup.com/Theory/rootkits.html

Miller, Toby. "Analysis of the T0rn rootkit". http://www.sans.org/y2k/t0rn.htm