

Contents

I	Theory Investigation	3
1	Introduction	4
1.1	File System Integrity	4
1.2	The place of FSI in the security model	4
1.3	Unix tools involved	5
1.3.1	ls	6
1.3.2	ps	6
1.3.3	netstat	6
1.3.4	strace	6
1.4	rootkits	6
1.5	checksums	6
2	How FSI works	8
2.1	The UNIX file system	8
2.2	FSI in a nutshell	8
2.2.1	Initialisation:	9
2.2.2	Verification:	10
2.2.3	Maintenance:	11
3	FSI tools available	12
3.1	Tripwire	12
3.2	AIDE	12
3.3	Samhain	13
3.4	Integrit	13
3.5	Intact	14
3.6	GFI LanGuard	14
II	Practical report	15
4	Introduction	16
4.1	Scenario 1	16
4.1.1	The attack	16
4.2	The administrators investigation	17
4.2.1	Basic troubleshooting	17
4.2.2	Deeper investigation	18
4.2.3	Finding what the attacker might have done	19
4.2.4	Conclusion	21
4.3	Installation, configuration and use of AIDE	21

4.3.1	The configuration file	22
5	Scenario 2	25
5.1	Alice exploits the same vulnerability	25
5.2	Bob trusts what the system says	26
5.3	Installation of Tripwire	26
6	Scenario 3	27
6.1	The attack	27
7	Improving FSI	28
7.1	Introduction	28
7.2	An unclean database	28
7.3	Remote checking	29
7.4	Initiate on a clean system in single mode user	30
7.5	More than one checksum	30
8	Installing Tripwire	32
8.1	Download and Installation:	32
8.2	Configuring the Policy	33
8.3	Initialise the Database	33
8.4	Checking the Filesystem	33
8.5	Updating Files	33
8.6	Updating the Policy	34
8.7	Scheduling Regular Checks	35
8.8	Protecting Web Content	35
III	References	36
IV	Appendices	38
	Appendix A - Post-Installation	A-39
	Appendix B - Tripwire Policy	B-42
	Appendix C - Tripwire Report	C-46
	Appendix D - Tripwire Database Update	D-50
	Appendix E - Updates to Tripwire Policy	E-56
	Appendix F - Sample UNIX Commands	F-58

Part I
Theory Investigation

Chapter 1

Introduction

1.1 File System Integrity

In the Internet-enabled age security is a major concern to system administrators, as soon as a system is connected to the Internet it is welcoming intruders to try break through the security¹. Figure 1.1 shows through the use of buffer overflows, it is possible to execute arbitrary code on servers. The chunk-encoding 'slapper' worm for the Apache webserver, or the Code Red worm for Microsoft IIS are both good examples of these. The scope of this report does not extend to buffer overflows. Of course, the entry to a system may simply be down to mis-configuration by the system administrator, or a poor password policy.

The motivation for a person (who should be referred to as 'attackers' or 'intruders' rather than 'Hackers') to gain access to systems which they are not authorised can be anywhere from:

- the same adrenaline as a joy-rider
- to deface
- stage a Distributed Denial of Service attack
- obtain personal information such as credit cards
- or simply for the sense of achievement.

Even if the attackers intentions are simply to gain entry but not cause damage, this and the other motivations are offences under the Computer Misuse Act 1990 [HMSO, 1990].

The motivation of attackers matters little the job of a system administrator is to keep them away! File System Integrity checkers play a crucial part in ensuring that nobody is lurking around the system.

1.2 The place of FSI in the security model

File System Integrity checks are not a weapon against intruders in the same way as a firewall is, they do nothing to stop an intruder compromising a system. Figure 1.1 shows

¹of course this applies equally to systems within the confines of an organisation.

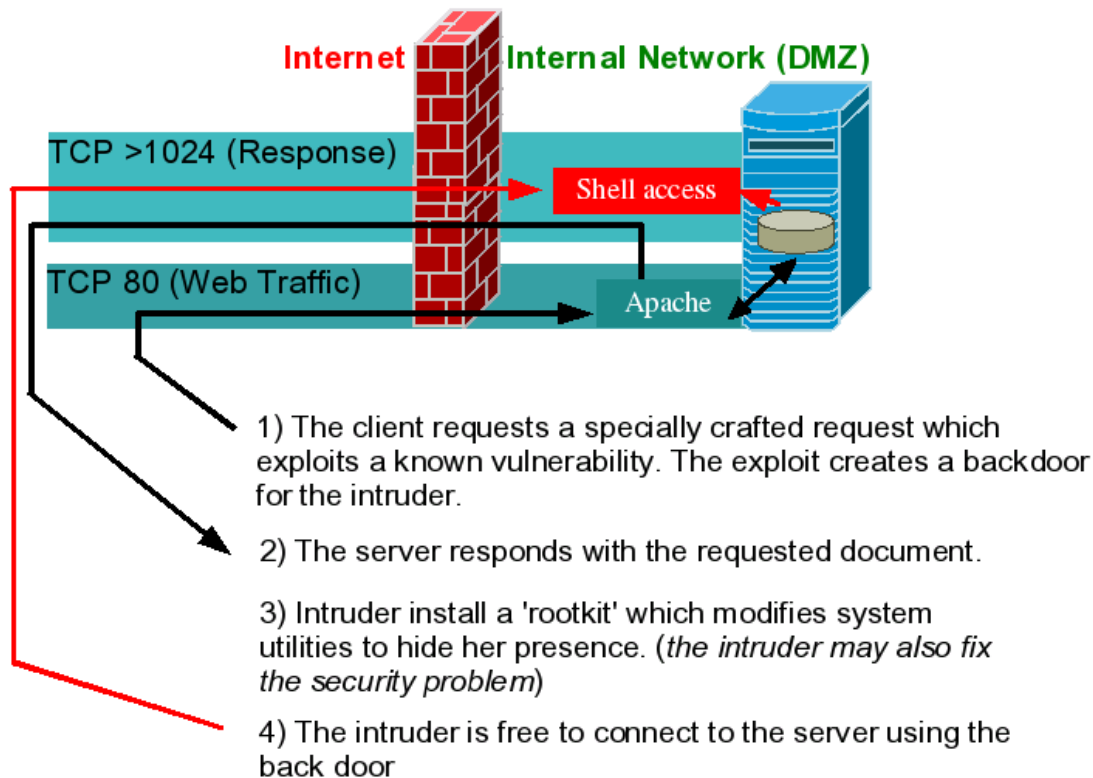


Figure 1.1: The diagram shows how a vulnerability in a running service (e.g. Apache) can be used to gain entry to a system.

entry is staged against a publicly running service— a tight set of firewall rules may help limit the damage— but will not stop entry through a buffer-overflow. FSI tools are to be used in conjunction with firewalls, chroot jails², and other intrusion detection tools such as portsentry³. In the event of a suspected breach FSI tools simply give system administrators confidence that crucial files within the operating system have not been compromised. It is important that if a system administrator suspects that a system has been compromised that there are conclusive logs available. With these logs the administrator can set about closing only the infected services whilst cleaning the damage— and minimising downtime for legitimate users. Without FSI tools the system administrator may incorrectly diagnose the problem and leave an infected system connected to the network, or incur a long period of downtime by re-installing the system from clean backup images.

1.3 Unix tools involved

This report deals with a number of Unix tools, if you are not familiar with the Unix operating system an overview of these commands can be found here (*Appendix F contains sample output from these commands.*)

²a chroot jail allows a service to run within it's own virtual file system, should an intrusion happen the damage is contained in a small isolated area of the file system.

³Portsentry is security software which detects intruders as they scan for services running on a server.

1.3.1 ls

ls list directory contents. This includes size, date of last change, owner, group, permissions and type of the file (file, socket, directory). [GNU ls manpage, 2002]

1.3.2 ps

ps gives a snapshot of the current [running] processes. [GNU ps manpage, 2002]

1.3.3 netstat

netstat prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. [GNU netstat manpage, 2000]

1.3.4 strace

strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them. Students, hackers and the overly-curious will find that a great deal can be learned about a system and its system calls by tracing even ordinary programs. And programmers will find that since system calls and signals are events that happen at the user/kernel interface, a close examination of this boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions. [GNU strace manpage, 1999]

1.4 rootkits

Should an intruder find a weakness in a system- and have the opportunity to execute any code they desire— the first choice would not be to wipe the disk, this choice isn't based out of compassion but a desire to hide and investigate what treats might be found deep inside the system. Should the server participate in Distributed Denial of Service attacks the damage will be traced to system administrators, and if the intruder did a good job- they won't be traced. Rootkits are widely available and simply consist of a number of modified tools, such as *ls*, *ps*, *netstat*, *who*, *md5sum*, *kill*.

By modifying the output of a directory listing, files which are infected can be hidden, any network connections to the intruder can be hidden, modifying the *kill* command so that it doesn't actually terminate rogue processes, and modifying *md5sum* ensures that it is difficult for the system administrator to detect trojan system utilities.

1.5 checksums

An important part of how File System Integrity tools work is by producing checksums of the file system. An example of a checksum used by all the tools mentioned in this report is the MD5 sum. The MD5 checksum is defined in RFC 1312 [RFC 1312, 1992]. MD5 sums are 128-bit checksums, and most often represented in the following form: **7d970df3bbf11a18170cad22b7e50814**.

Given a small amount of effort for today's computing power— it is not difficult for a rootkit to create binaries which compute the same MD5 checksum as the original. By

using additional checksums such as SHA-1, HAVAL and RIPE-MD160 the complexity to create binaries which maintain correct checksums for all message digest algorithms is computationally more secure.

Chapter 2

How FSI works

2.1 The UNIX file system

To help understand FSI tools it useful to understand the underlying file system. This report concentrates on the UNIX file system, since it was not until recently that Microsoft encouraged users to use a file system with in-built security.

The following information refers to the EXT-2 file system:

Every file or directory on the file system is described by **exactly one** inode. Inodes contain the following information:

- **mode** is the item a file, directory, symbolic link or device
- **ownership** who owns the file
- **size** of the file in bytes
- **timestamp** the time the item was created, last modified and last accessed.
- **datablocks** pointers to the physical blocks of the disk containing the item
- **id** each inode has an identification number

2.2 FSI in a nutshell

In a nutshell file system integrity works in the following manner:

- Create a baseline image of signatures of system utilities using hash algorithms such as RSA's MD-5.
- At scheduled intervals (*or if a intrusion is suspected*) check the live system against the baseline image.
- Unless the system administrator has updated a package the signature of system utilities or configuration files should not have changed. (*It is required that system administrators update the baseline image after updating utilities.*)

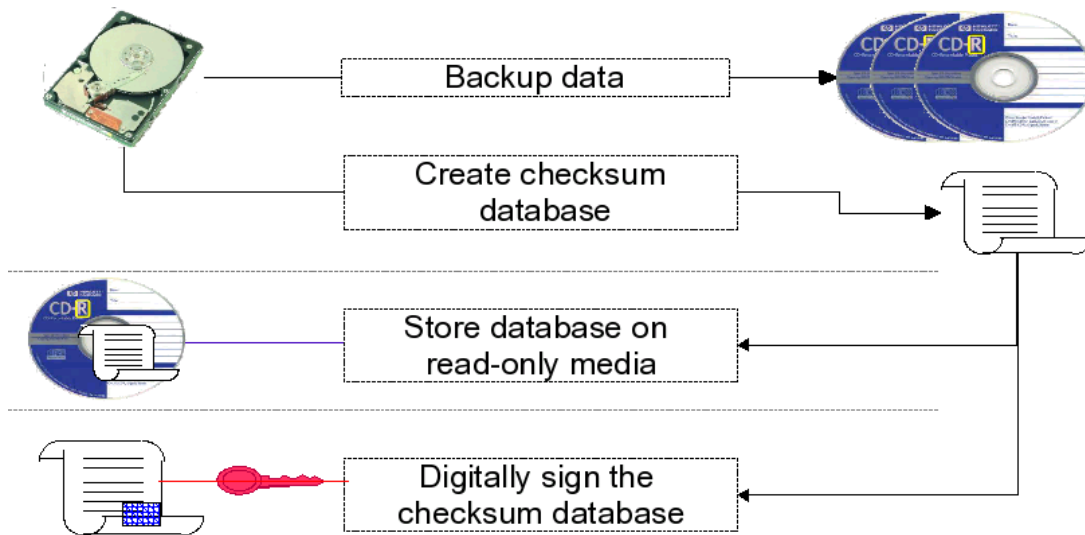


Figure 2.1: The diagram shows the initialisation process of File System Integrity tools.

- Unless log files have been rotated they should not shrink, but grow in size over time
- System directories such as `/root` should not have files modified.

2.2.1 Initialisation:

Using FSI tools on a system is a decision that need to be made at the beginning stages— as soon as a system is connected to a network it is potentially vulnerable to compromise. The most important stage of deploying FSI tools is creating the database of checksums. Figure 2.1 shows the initialisation process of FSI tools.

- Create backup images of the system (*while missing this stage does not hinder FSI tools themselves— it makes sense to have the ability to quickly recover files.*)
- Define a FSI policy (*A sample tripwire policy is included in Appendix B*), which will decide which files, and attributes should be checked. Files may be defined in a number of categories as:
 - **Critical** - files that cannot change
 - **Binaries** - files the should not change
 - **Config** - config files that are changed infrequently
 - **Log** - files that should grow, but not change
 - **Invariant** - files that should never change permission or ownership
- Create a database of checksums:
 During the initialisation phase the FSI tool will search through system binaries, and configuration files creating checksums (*Tripwire for example uses MD5, whilst AIDE uses multiple checksums.*). As well as computing the checksum for a file specific file system information is stored.

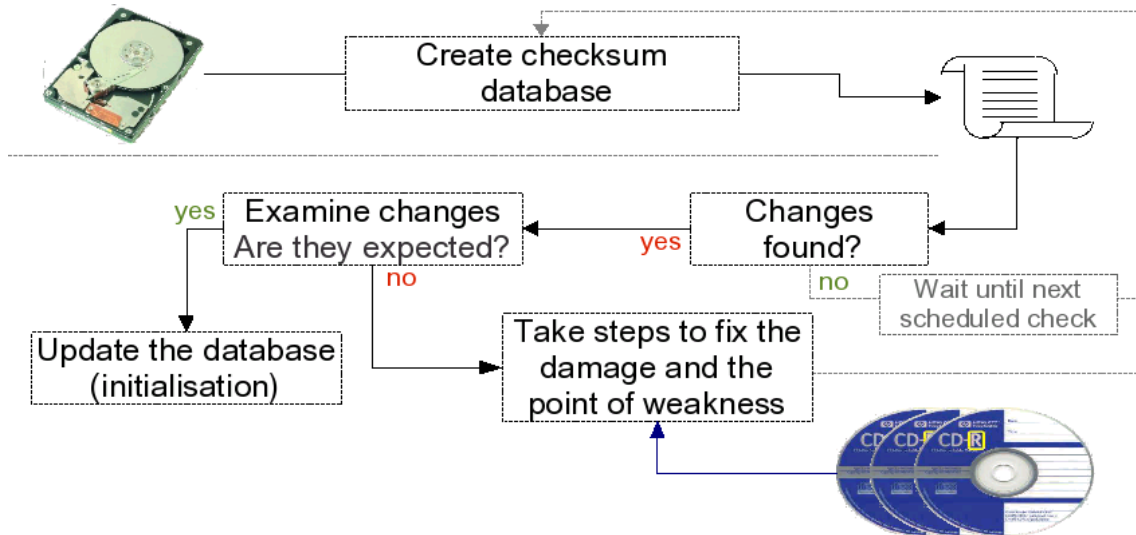


Figure 2.2: The diagram shows the verification process of File System Integrity tools.

- Depending on the FSI tool used one of the following steps is required to prevent an intruder from manipulating the FSI tool:
 - The traditional method is to keep the FSI tool secure was to move the database onto read-only media for protection. If this is a CD-R this makes updates tedious; however this could also be a remote-server¹
 - To ease administration, tools such as Tripwire support digitally signing the database of checksums with public-key encryption².
- Finally the administrator of the system should run a manual check using the FSI tool- to ensure that the FSI tool has been correctly configured.

2.2.2 Verification:

The FSI tool should be run at regular intervals to ensure that the file system has not been tampered with. Since FSI checking can be a disk and CPU intensive task, it may not be desirable to run during office hours. (*Appendix C contains a sample Tripwire report*³.) Figure 2.2 depicts the process of verification:

- by re-creating the database of checksums from the file system in the current condition then if the original database **is different** from the current database of checksums
- Carefully analyse the report
 - Have any packages been upgraded and the checksum database was not updated
 - Have log files been rotated recently

¹although it would be important to be sure that the remote host had not been compromised.

²This is a TODO for the AIDE project.

³The report shows that a number of files have changed as part of the NVidia, Gnomemeeting, VNC and Shadowutils packages.

- Is the FSI tool too sensitive (*it may be necessary to update the policy*)
- Is there a genuine security breach
- If there are genuine security concerns then:
 - Disconnect the system from the network
 - *If required* create a backup of the system to aid examination into how the intrusion happened⁴.
 - Replace the infected files from backup (*If a backup was not created it may be necessary to re-install the operating system*)
 - Re-compute the database of checksums— ensuring that the compromised files pass the cryptographic checks (*it should be expected that they will fail the last-modified check for the obvious reason that they were restored from the database.*)

2.2.3 Maintenance:

From time to time we need to maintain the system, since a key element of security is keeping software up-to date. The damage caused by the Code Red worm— would have been limited if system administrators understood how critical applying patches is in maintaining system security.

A trivial example is a recent discovery that the VNC client could use the same authentication challenge⁵— with the potential for replay attacks.

It is critical that we update our VNC binaries, but they are protected by the FSI tool— by updating the insecure package a warning is introduced by the FSI tool. We need to maintain the database from time-to-time and ensure that it remains current.

Appendix D includes a sample Tripwire update report (*as a result of modifying Apache configuration and starting the web-server for the first time.*).

The report shows the expected details drawn from the database, as well as the new values— allowing the system administrator to be 100% certain that the changes are correct. All that remains is to updated the checksum database to include the changed details.

⁴The infected files may be used as part of a honey-pot— to trap the intruder should they return on an isolated server.

⁵<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1336>

Chapter 3

FSI tools available

In the field of file system integrity, there is a leader— Tripwire started as proprietary software, however over 2 years ago Tripwire Inc released the core of Tripwire under the General Public Licence. Tripwire Inc's commercial offering of Tripwire includes additional utilities to ease administration, as well as providing support for the software.

Many of the other programs are open-source programs, some developed as free replacements of Tripwire¹. Because of the nature of the GPL licence there are many File System Integrity checkers, many are 'different' versions of the common programs.

Rather than listing every FSI tool available this report details the most popular tools available. The selection of programs run on Unix / Mac OS/X² / Microsoft Windows³.

3.1 Tripwire

Tripwire is known for it's high reliability and effectiveness and has been around for a number of years. Tripwire works by taking a digital snapshot (baseline image) of a system in it's optimal, known good state. It then compares the system against the digital snapshot and identifies any deviations. The digital snapshot is protected by cryptography with many features to ensure that once an attacker has gained access to your system- that they cannot hide. The digital snapshot of the files is created using the MD5 checksum.

<http://www.tripwire.com/> (Proprietary version)

<http://www.tripwire.org/> (Free version)

OS: GNU/Linux, Solaris, *BSD and other Unices.

The proprietary version also runs on MS Windows based on NT.

Licences: GNU General Public Licence and proprietary licence

3.2 AIDE

AIDE is an open-source replacement for Tripwire, the design goals of AIDE was to be a completely free replacement of Tripwire, which surpassed the limitations of Tripwire.

¹When we say a program is free, we mean that it is released under a free software licence such as the GNU GPL or any opensource compatible licence

²Most of them have been or can be ported to Mac OS/X- a POSIX compliant OS)

³Tripwire exists for Windows, but it should be no problem to use the others thanks to Cygwin- software that gives facilities for running a Unix environment under Windows

However the disadvantage of AIDE is that users rely on community support rather than a commercial entity.

AIDE creates a database of files which are specified in the config file. The database stores file attributes (permissions, inode number, owner, file size, modification time, creation time and access time, growing size, and number of links). AIDE takes the checksum a little further than Tripwire and creates cryptographic checksums based on several message digest algorithms (MD5, SHA1, RMD160, TIGER, HAVEL).

<http://www.cs.tut.fi/~rammer/aide.html>

OS: GNU/Linux, Solaris, *BSD and other Unices.

Licence: GNU General Public Licence

3.3 Samhain

Samhain is a File System Integrity checker which can be used on both single hosts as well as networks. Provisions of Samhain are to provide details of changes made and who was logged in at the time of changes. Samhain searches the file system for SUID binaries⁴. However the main feature which sets Samhain apart from any other FSI checker is detection for kernel rootkits.

A further feature of Samhain is the ability to centralise monitoring- by taking advantage of an encrypted client/server architecture.

Like other File System Integrity checkers Samhain uses cryptographic checksums of files to ensure that no modifications have taken place.

<http://la-samhna.de/samhain/>

OS: GNU/Linux, Solaris, *BSD and other Unices.

Licence: GNU General Public Licence

3.4 Integrit

Integrit brings a number of features to the table over other FSI checking tools. The advantages are a small memory footprint during runtime (ensuring the system can carry on with it's intended task). An option to reset the access times of files are producing checksums.

Integrit allows:

- **Small memory footprint during runtime.** This is a big deal because a machine that is important enough to protect is probably doing important things.
- An **option to reset the access times** of selected files or directory trees after doing checksums
- Output format can be **XML** or an **easy-to-scan human-readable** format

⁴SUID binaries are programs which users are able execute with root, or Administrator privileges.

- **Simultaneous check and update:** integrity can generate a new database while running a check against an old database

<http://integrity.sourceforge.net/>

OS: GNU/Linux, Solaris, *BSD and other Unices.

Licence: GNU General Public Licence

3.5 Intact

Unlike the other FSI checking tools listed above which are scheduled tools, Intact has the benefit that after creating the image of the filesystem- it monitors the changes in real-time.

<http://www.pedestalsoftware.com/intact/>

OS: Microsoft Windows NT, Solaris and GNU/Linux.

Licence: proprietary licence.

3.6 GFI LanGuard

GFI LanGuard works much like other FSI tools, by holding a database of MD5 checksums, and at regular intervals re-computing the checksums to ensure that the files have not been tampered with.

<http://www.gfi.com/>

OS: Microsoft Windows 2000/XP

Licences: Freeware

Part II

Practical report

Chapter 4

Introduction

We will present here three scenarios in which the integrity of the file system may have been compromised.

Bob is the system administrator for his company and Alice is an attacker. Bob has set up a firewall for his company, and he believes that he can blindly rely on it for his network security.

4.1 Scenario 1

4.1.1 The attack

Alice, finds a security hole that allows her to exploit one of Bob's machines. Alice now has access to the machine and will be free to do whatever she decides:

- **Replaces the files:**

Alice replaces the system binaries with corrupted versions so she can hide her activity.

- **Remove her fingerprints:**

The log files are the place where the system administrator could find evidence of an intrusion— that is why Alice will:

- **Remove** the relevant lines already in the log files
- Set the log file to **read-only mode**, so that the system logger will not write anymore information

- **Block the security hole and install a backdoor:**

Of course, Alice does not want someone else to have the opportunity to take advantage of the security hole. She will patch the vulnerability, whilst installing a backdoor to allow access to the machine.

- **Download and install the rootkit:**

Alice will download a rootkit from the Internet. Alice will then be able to replace some binaries that Bob would use to investigate if the machine has been compromised. Each binary of the rootkit actually uses a configuration file to determine how the forged binaries should behave. Alice configures the rootkit like this:

- 'ls' should not show the binary of the backdoor and configuration files of the rootkit
- 'ps' should not show the processes run by the Alice (the backdoors' process, or any processes executed by Alice)
- 'netstat' should not show Alice's connection within the backdoor

- **Finishing the work:**

Alice turns the log files back to read-write mode. She is now able to gain access to the machine without any password. Her activity on the machine is being hidden by the forged binaries from the rootkit.

4.2 The administrators investigation

Bob discovers from log files on the server that there was a portscan originating from an external host. Bob believes that there should only be connections from the internal LAN. Bob decides that it would be a good idea to ensure that there has been no other un-authorized access.

4.2.1 Basic troubleshooting

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 143.52.2.4:ssh         143.52.2.34:1234      ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags               Type                   State                  I-Node Path
unix    9      [ ]                 DGRAM                  163                    /dev/log
unix    3      [ ]                 STREAM                 CONNECTED              8505                    /tmp/.X11-unix/X0
unix    3      [ ]                 STREAM                 CONNECTED              8504
```

Bob's first task is to check for connections to and from the server. Everything appears as Bob expects— there are no suspicious connections from the external world listed. (*The ssh connection is coming from Bob's workstation, because Bob was checking remotely to determine the state of the system.*)

```
root pts/2 Feb 25 21:20 (bob-admin.company.com)
```

The *who* command only shows Bob's current connection.

Bob now decides to check the running processes on the system with the *ps* command:

```
USER  PID  %CPU  %MEM  VSZ  RSS  TTY  STAT  START  TIME  COMMAND
root   1    0.0  0.3 1220  420 ?    S    10:42  0:03  init [2]
root   2    0.0  0.0   0    0 ?    SW   10:42  0:00  [keventd]
root   3    0.0  0.0   0    0 ?    SWN  10:42  0:00  [ksoftirqd_CPU0]
root   4    0.0  0.0   0    0 ?    SW   10:42  0:01  [kswapd]
```

```

root    5  0.0  0.0    0    0 ?    SW  10:42 0:00 [bdf flush]
root    6  0.0  0.0    0    0 ?    SW  10:42 0:04 [kupdated]
root    7  0.0  0.0    0    0 ?    SW  10:42 0:00 [i2oevtd]
root    8  0.0  0.0    0    0 ?    SW  10:42 0:00 [kreiserfsd]
root   34  0.0  0.0    0    0 ?    SW  10:43 0:00 [khubd]
root   509 0.0  0.4  1288  540 ?    S   10:43 0:00 /sbin/syslogd
root   512 0.0  0.3  2048  396 ?    S   10:43 0:00 /sbin/klogd
root   594 0.0  0.4  2504  528 ?    S   10:43 0:00 /usr/sbin/sshd
root   619 0.0  0.4  1396  584 ?    S   10:43 0:00 /usr/sbin/cron
root   930 0.0  0.2  1204  372 tty1  S   10:43 0:00 /sbin/getty 38400 tty1
root  5568 0.0  1.0  2320 1304 pts/0  S   14:22 0:00 bash
root  6543 0.0  1.2  3480 1612 pts/0  R   15:05 0:00 ps aux

```

The few running processes seem to reflect normal activity of the server— Bob does not find any process he did not expect to find running on the server.

Bob believes that the machine was not compromised, and the portscan detected was simply a script-kiddie¹. Bob has received reports that the server has seemed un-responsive, and observes that there is more hard-drive activity than expected. Bob checks again with *ps* to see if there are any scheduled background tasks running, or anything that might explain the machine's behaviour but NOTHING !

4.2.2 Deeper investigation

After an hour of troubleshooting, Bob decides to check with *strace* if there are any nasty things happening with the *ps* command.

The output from *strace* below has been partially stripped to highlight the interesting section.

```

machine:~# strace ls >/dev/null
execve("/bin/ls", ["ls"], [/* 13 vars */]) = 0
uname({sys="Linux", node="foobar", ...}) = 0
brk(0) = 0x8054b94
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=32197, ...}) = 0
old_mmap(NULL, 32197, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40012000
close(3) = 0
open("/lib/libc.so.6", O_RDONLY) = 3
[...]
open("/dev/ptyxx/.file", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=18, ...}) = 0
[...]
write(1, "foo"... , 40) = 40
close(1) = 0

```

¹An in-experienced user who believes that they are 'hackers'— but only possess a small amount of knowledge to replicate others code

```
munmap(0x40012000, 4096)           = 0
semget(IPC_PRIVATE, 1074951168, 0) = -1 ENOSYS (Function not implemented)
_exit(0)                          = ?
```

Here Bob, finds in the output something that he finds suspicious:

```
open("/dev/ptyxx/.file", O_RDONLY|O_LARGEFILE) = 3
```

/dev/ptyxx is not expected behaviour— Bob wonders what could be in that directory. Bob will now check other binaries with *strace*.

- **syslogd**

```
foobar:~# strace syslogd >/dev/null
[...]
open("/dev/ptyxx/.log", O_RDONLY|O_LARGEFILE) = 3
[...]
```

- **ps**

```
foobar:~# strace ps aux >/dev/null
[...]
open("/dev/ptyxx/.proc", O_RDONLY|O_LARGEFILE) = 3
[...]
```

- **netstat**

```
foobar:~# strace netstat >/dev/null
[...]
open("/dev/ptyxx/.addr", O_RDONLY|O_LARGEFILE) = 3
[...]
```

4.2.3 Finding what the attacker might have done

There are unexpected files in the directory */dev/ptyxx* (*which is a directory that is not created on installation of the UNIX operating system.*). Executing *ls* in that directory shows nothing— Bob now suspects the *ls* command to be compromised. Bob then uses *more* to view the suspicious files:

```
foobar:~# more /dev/ptyxx/.file
.file
.log
.proc
.addr
[...]
alice_ssh
alice_telnet
alice_nmap
backdoor
```

Bob now has evidences and concludes the following:

- The machine has been compromised
- We have no idea what damage has been done— and can't figure it out
- We can not trust the system anymore
- We need to reinstall the system from scratch.

Bob now disconnects the machine from the network and will investigate the malicious files that might help him figuring out what the attacker has done.

- **ls and /dev/ptyxx/.file**

Bob has seen the content of */dev/ptyxx/.file* and begins to understand— this file looks like a configuration file for the forged *ls* from the rootkit. Basically, what it does is to tell *ls* which files should be hidden in the output of the command. For example *ls* will hide the configuration files of the rootkit's forged command but to also hide the files: *backdoor*, *alice_ssh*, *alice_telnet*, *alice_nmap*.

- **syslogd and /dev/ptyxx/.log**

```
foobar:~# more /dev/ptyxx/.log
port 1234
[...]
from 193.12.34.56
from 193.56.12.34
[...]
```

Viewing the *.log* file— Bob realises that the syslogger was not logging all activity. Alice, has taken care of ensuring the syslogger does not write any of her connection to the logs. Bob can be sure than an intruder was connecting using these IP addresses and port.

- **syslogd and /dev/ptyxx/.log**

```
foobar:~# more /dev/ptyxx/.addr
3 1234
4 1122
4 1123
4 1124
[...]
2 193.12.34.56
2 193.56.12.34
[...]
```

Here again, Bob guesses that the attacker was willing to hide all of his connection. So, *netstat* will show everything but the connections related to the attacker's activity.

- **ps and /dev/ptyxx/.proc**

```
foobbar:~# more /dev/ptyxx/.proc
2 alice\_ssh
2 alice\_telnet
2 alice\_nmap
[...]
2 backdoor
[...]
```

Bob now comes to the realisation that when he used the *ps* command, it was also hiding processes called *alice_ssh*, *alice_telnet*, *alice_nmap* or *backdoor*.

4.2.4 Conclusion

Bob already knows that the machine has been compromised, but also has evidence that someone was using it. It is obvious that Alice connects from the source IP address *193.12.34.56* to the port *1234* using a backdoor— the binary called *backdoor*. Also, Alice was using some other programs to get access to other machines on the internal network. The use of the nmap utility² by Alice was also masked by the forged *ps* command.

Re-installing a clean system takes time, although the damage in this scenario was not great— however the story could have been very different. It would have been possible for the machine to be participating in a Distributed Denial of Service attack, or sensitive information (such as credit card details or personal employee information) could have been viewed by the intruder. In this scenario there is no way to be sure what damage was caused, or what data may have been changed.

Bob does not want the system to be compromised again— so decides to implement a File System Integrity tool to provide a method of checking the integrity of the files.

4.3 Installation, configuration and use of AIDE

Bob has installed the File System Integrity tool, AIDE and now has the task of configuring the file checker.

- **p permissions:** read, write and execution for the user, group and others
- **i inode:** the inode of the file system
- **n number of links:** number of hard or symbolic links to that file
- **u user:** the owner of the file
- **g group:** the group that file has been assigned to
- **s size**

²Nmap (Network Mapper) scans hosts on a network for open TCP or UDP ports.

- **b block count**
- **m mtime**: last modified time
- **a atime**: last accessed time
- **c ctime**: last changed time
- **S check for growing size**
- **md5 md5 checksum**
- **sha1 sha1 checksum**

4.3.1 The configuration file

```
# AIDE conf

database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new
gzip_dbout=yes

# Custom rules
Binlib = p+i+n+u+g+s+b+m+c+md5+sha1
ConfFiles = p+i+n+u+g+s+b+m+c+md5+sha1
Logs = p+i+n+u+g+S
Devices = p+i+n+u+g+s+b+c+md5+sha1

# What directories/files we want in the database

# Kernel, system map, etc.
=/boot\$ Binlib
# Binaries
/bin Binlib
/sbin Binlib
/usr/bin Binlib
/usr/sbin Binlib

# Libraries
/lib Binlib

# Devices
!/dev/pts
/dev Devices

# Logs
/var/log Logs
```

In this configuration, Bob has defined some rules. For example:
Binlib = p+i+n+u+g+s+b+m+c+md5+sha1
 means to check for permissions, inode, number of links, user, group, size, block count,

modification time, change time, md5 checksum and sha1 checksum.

Then Bob can apply this to an entire directory such as */sbin*.

This configuration will check the integrity of some directories containing binaries, libraries, devices or logs.

- **Initialisation**

Once Bob has defined what he wants to check for integrity he needs to generate a database of it with the command *aide -init*.

- **Usage**

On a regular basis Bob will generate a new database and compare it to the initial database. The command *aide -update* will generate this new database and automatically compare the two databases.

It is obvious that defining the perfect configuration file for a system is not easy. At first the database should be checked by running an initial update to the database. Some files will naturally change on the file system, the configuration needs to be fine tuned until it is acceptable. An example of the natural changes can be seen below, with a report generated by *aide -update* after the database had first been generated:

```
AIDE found differences between database and file system!!
Start timestamp: 2003-02-24 00:38:02
Summary:
Total number of files=7241,added files=0,removed files=0,changed files=3

Changed files:
changed:/dev/tty3
changed:/dev/pts/5
changed:/dev/xconsole
Detailed information about changes:
```

```
File: /dev/tty3
  Uid      : 0                , 1000
  Gid      : 0                , 5
  Ctime    : 2003-02-23 09:12:07 , 2003-02-23 15:32:03
File: /dev/pts/5
  Uid      : 0                , 1000
  Gid      : 0                , 5
  Ctime    : 2003-02-23 09:12:07 , 2003-02-24 00:12:36
File: /dev/xconsole
  Ctime    : 2003-02-23 15:18:50 , 2003-02-24 00:12:36
```

Some files have already changed but this does not mean that Alice is already back trying to compromise the system. Those files are devices— relating to virtual terminals of the system. When Bob establishes his ssh connection to the system to check the file system integrity— the ssh session is assigned a tty device: *e.g*, */dev/pts/5*. Each key-stroke that Bob enters in his virtual terminal will cause the device file */dev/pts/5* to be modified. This activity is a normal part of the Unix architecture.

The report shows that the files have been changed, it tells the time of the last change –versus– the last modification time which is stored in the database.

Bob is able to modify the configuration file to avoid this and overwrite the initial database, with *aide -update*.

Chapter 5

Scenario 2

Alice is not the kind of person who gives up easily, she has noticed that she cannot access the machine through her backdoor. Alice guesses that the system administrator has discovered her previous attack and reinstalled everything from scratch.

The machine is now making use of the AIDE file system integrity checker, however Bob working under the stress of having to return the system to the network forgot to update the service which had the security hole. Alice is now free to exploit the same vulnerability as in the previous scenario— and take control of the machine again.

5.1 Alice exploits the same vulnerability

- **Exploit and rootkit installation**

Alice will use the same vulnerability to compromise the system again, she will follow the same steps:

- Replacement of the files
- Removal of her fingerprints
- Block the security hole and install a backdoor
- Download and install the rootkit

- **Bypass the FSI checker**

Alice notices that the system administrator has installed AIDE to check the file system integrity and knows that her presence will be detected as soon as the check will be performed. Alice reads the configuration file to find out where the database is located and finds out that the storage of this database is a directory on the hard disk. Since this media is read write, Alice laughs out loud— and then proceeds to:

- Stop the cron daemon to avoid AIDE automatically executing, and gain precious time
- Take time to study the configuration and find where the FSI checker does not check
- Install the rootkit
- Regenerate the database

- Restart the cron daemon

Alice has been able to bypass the FSI checker because the database was on read/write media. This means that she could very easily regenerate the base after the rootkit was installed. The situation in Scenario 2 is worse for the system administrator—since he will believe that the integrity of the file system is not in question.

5.2 Bob trusts what the system says

Bob receives complaints from his users telling that the network is extremely slow. After he checks logs on different machines he sees that there are still a number of portscan originating from the server which is protected by AIDE. Bob uses the *aide -check* command to check if the system has been attacked. Bob knows that any compromised binaries using will be detected by the file system integrity tool.

Bob first checks for connections from outside and unexpected processes running— but he can't find anything. Bob thinks that the file system might have been compromised. Bob starts a check on the file system and his surprised to see that AIDE has not detected any compromised files.

Bob continues the investigation and finally understands that the database had been regenerated by the attacker. Bob now understand's that the FSI tool can be more harmful if it is not configured correctly— if only Bob had stored the database on read-only media such as a CD-Rom this latest attack might have been discovered much quicker— **and if only Bob had updated the vulnerable service then this second attack would not have happened.**

5.3 Installation of Tripwire

Since Bob prefers to work from his own system rather than walk to the server room¹, Bob decides on using Tripwire. Tripwire uses public-key encryption and digital signatures to protect the database— rather than having to physically attend the server when updates are required. (Installation instructions for Tripwire on RedHat Linux are included in the chapter— Installing Tripwire)

Bob will reinstall the system and update the service that had a security hole.

¹For servers which are co-located this is a big limitation— especially if the server resides in a different country

Chapter 6

Scenario 3

A few months later, a security advisory is published [Redhat Network a, 2002] after a security hole has been found in the Apache web server. Alice knows that Apache was running on the machine she attacked a few months ago and wonders whether the admin has already updated the package or not.

6.1 The attack

It does not take more than 5 minutes for Alice to get into that computer and she wonders now if AIDE is still running with a database on read/write media. First Alice stops the cron daemon to make sure that no automated job will report a modification of the system and then have more time to investigate the machine. She finds out that Bob has replaced AIDE by Tripwire. **Alice is stuck!**

Before Alice can make changes to the Tripwire configuration or database she will need to already know Bob's passphrase to unlock his private key. It would be computationally expensive to bruteforce Bob's passphrase. Unless Bob has a weak passphrase which Alice guesses - **there is no way for Alice to modify the current installation of Tripwire.** Alice could remove the current installation and start over with a tripwire— which will ignore her activity. However by generating a new set of keys, whenever Bob comes to administrate the Tripwire installation— he will be locked out.

It is clear that if Alice tries to modify or re-install the Tripwire installation— Bob will be alerted that there are problems when he administrates Tripwire.

Alice has her traces (*where she connected from*) in the log files but she can't remove them because the FSI checker may detect the log file has reduced in size. If Alice installs a rootkit, it will be detected. Alice does not want to be caught, and knows she has no chance to compromise the system without being noticed. Alice decides to remove the traces from the logs— although the system administrator will see that someone has tried to compromise the system, they will have no evidence.

Chapter 7

Improving FSI

7.1 Introduction

Through the scenarios in the previous chapter the use of File System Integrity has been explored. It is clear to see that badly configured FSI tools are worse than no FSI tools. There are a few additional hints listed to maintain File System Integrity. Even though Tripwire uses digital signatures to protect the policy file and database—these points are still valid.

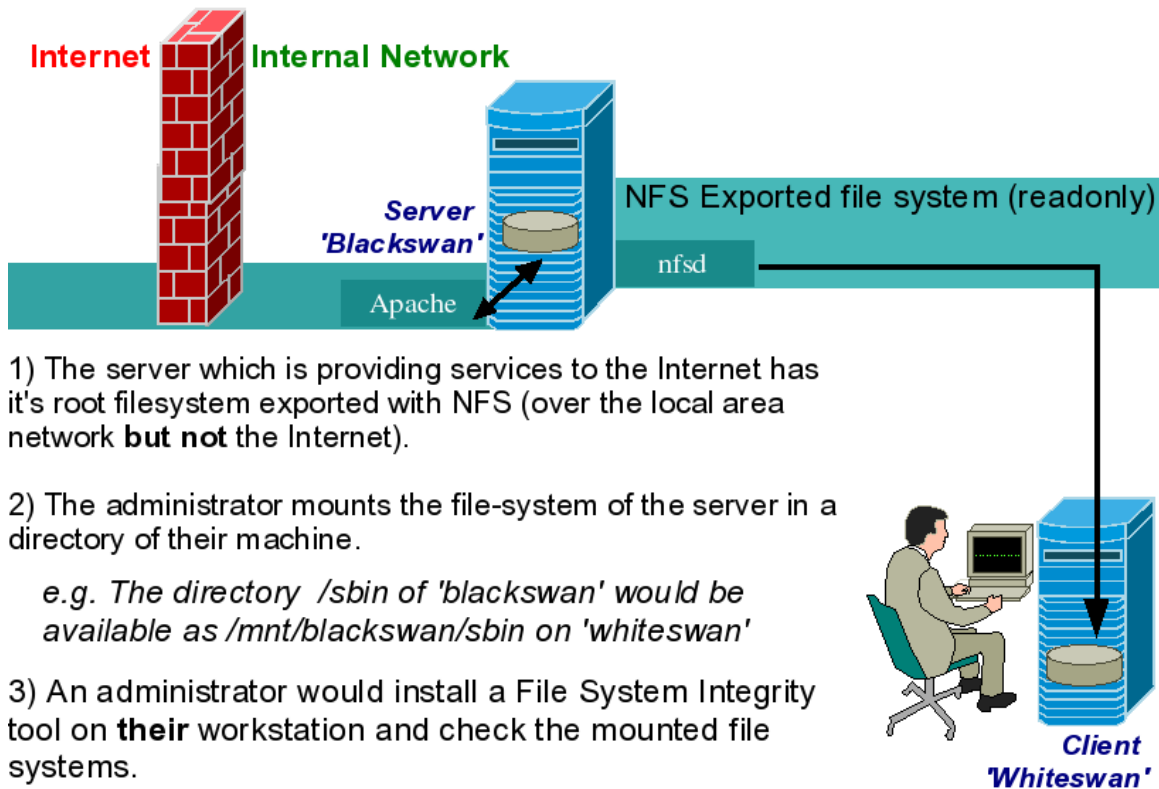
7.2 An unclean database

Imagine that you have generated the database of the files, you check the integrity of the file system— which will tell you that nothing has changed. If an attacker installs a rootkit and then generates a new database, it will also say that there is no difference between the baseline and the system. This what happened in the scenario 1.

Imagine that after you generate the database you explicitly change something on the system. For example, if you add a file in the directory */sbin*, this would have the following consequences:

- 1 added file in */sbin* (*with the current access, modification and creation times.*)
- 1 change to access time of */sbin*
- 1 change to modification time of */sbin*

The difference here is that the system administrator purposely created a violation in the database. If the attacker was to re-create the database they would need to also ensure that they follow the same steps as the administrator to create the violation. While technically not difficult to modify times— **understanding what the system administrator is expecting a particular violation is very difficult.**



1) The server which is providing services to the Internet has its root filesystem exported with NFS (over the local area network **but not the Internet**).

2) The administrator mounts the file-system of the server in a directory of their machine.

e.g. The directory /sbin of 'blackswan' would be available as /mnt/blackswan/sbin on 'whiteswan'

3) An administrator would install a File System Integrity tool on **their** workstation and check the mounted file systems.

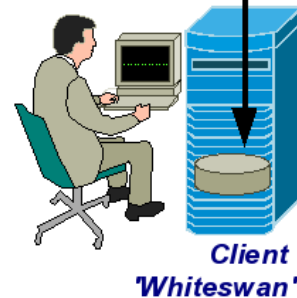


Figure 7.1: diagram to illustrate the process of remote file system integrity checking.

7.3 Remote checking

Alice could have compromised the binary of tripwire that checks for integrity¹ in order to make tripwire ignore changes an attacker makes to the system.

The best way to make sure that the FSI checker itself will not be compromised would be to designate a machine for the purpose of checking file system integrity (*See Figure 7.1*). The host checking FSI will not be connected to the Internet— so it remains secure. By using NFS² to mount the filesystem of the host providing external services (*blackswan*) in a directory on the FSI checking machine (*whiteswan*).

Unless the attacker attacks the machine we use to check the system we can have confidence in the Tripwire binaries.

Even if the attacker determines that the filesystem can be remotely exported— the attacker does not know that the system has its file system integrity checked remotely.

An extension to the idea of remote checking, would be to store a clean copy of the binaries on the client (*machine responsible for checking the MD5-sum*). Processing reports through Unix utilities such as awk, it would be possible to update binaries using utilities such as *rsync*. Although one should question if they are comfortable with the idea of updating critical system binaries over the network.

¹there are couple of rootkit for tripwire on the internet that can easily be customised to behave so that they would not show activity of an attacker.

²NFS allows directories *or the entire filesystem* to be transparently exported over network shares.

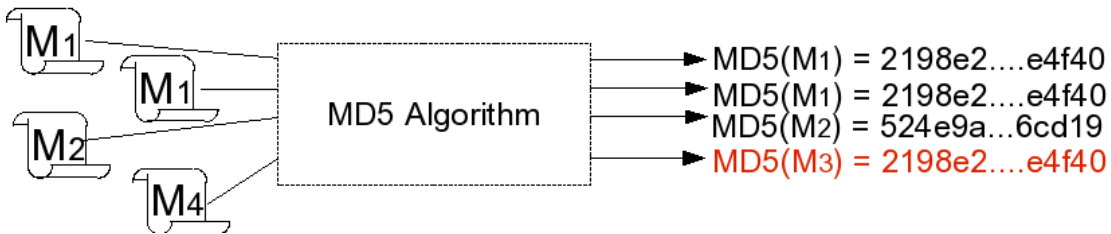


Figure 7.2: diagram to illustrate that it is possible for two different files to have the same resulting MD5 checksum.

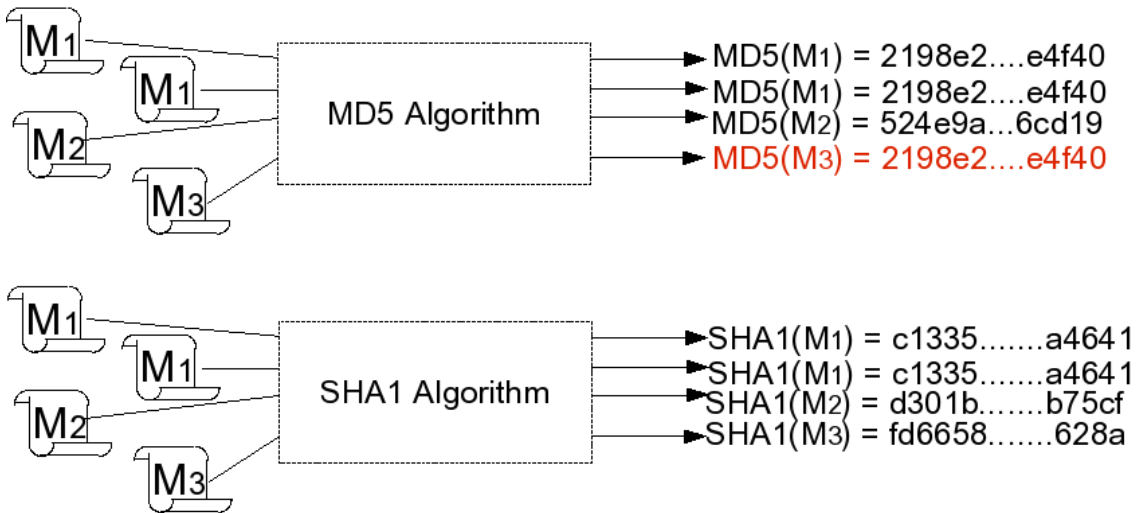


Figure 7.3: diagram to illustrate combining two different checksum algorithms eliminates the problems of checksum collisions.

7.4 Initiate on a clean system in single mode user

Another important point is to install the FSI checker on a clean system- immediately after it has been installed. The system should not have been connected to an untrusted network in the past. The file system integrity checker should be completed in single mode user for the installation.

7.5 More than one checksum

In the introduction rootkit's were introduced which were able to create the same MD5 checksum as the binary— although computationally expensive it is possible. In recent time the suitability of MD5 has been called into question, "There is a good chance that among the hash values of $2^{(n/2)}$ different messages a collision occurs. To prevent an attacker from finding collisions simply by computing that many values (birthday attack) n should be large enough." [Dobbertin H, 1996]. Figure 7.2 illustrates the possibility for collisions within the MD5 checksum algorithm.

By introducing a second checksum such as SHA-1 [RFC 3174, 2001]— we are reducing the number of collisions that an attacker could take advantage of, since the collision would

need to be present in both the MD5 and SHA-1 algorithms.

AIDE already uses multiple checksums, whilst there appears to be support for this in Tripwire— but it is not enabled by default. Figure 7.3 illustrates the improvement of using both MD5 and SHA-1 as checksum— however creating a second checksum produces twice as much strain on the CPU both when checking and initialising the database.

Chapter 8

Installing Tripwire

The following chapter describes the installation of the GPL'd version of the Tripwire file system integrity tool **Tripwire** for RedHat Linux¹. Tripwire requires super-user privileges— Tripwire will not provide meaningful results if it cannot read the files that it should be protecting.

8.1 Download and Installation:

- The first step is to download Tripwire— this can be downloaded for free². Tripwire binaries can be downloaded from <http://www.tripwire.org> or the source from <http://www.sourceforge.net/projects/tripwire/>.
- The RPM file downloaded from Tripwire will be contained in a *.tar.gz* file— this can be extracted with the command:

```
tar xvfz tripwire-2.3.xx.i386.tar.gz
```

- Now you will need to install the RPM with the command:

```
rpm -ivh tripwire-2.3-xx.i386.rpm
```

- Unfortunately, the installation of the RPM does not complete all the required aspects of installation. After installing the RPM the post-installation script needs to be executed: */etc/tripwire/install.sh* The output of the post-installation is included in Appendix A.

During the post-install there is the need for two passphrases— the **local** and **site** passphrases. These passphrases are used to digitally sign the database (*local*) and configuration files (*site*) with public/private keys. It goes without saying that it is important to choose appropriate passwords. **It is important to remember your site and local passphrases- as there is *no* provision for password recovery.**

¹These instructions are based around RedHat 7.3 - Valhalla. The instructions should transpose across to any RPM base Linux distribution.

²under the terms of the GPL- there are however restrictions on re-distribution.

8.2 Configuring the Policy

The default policy shipped with Tripwire is suitable for RedHat 7.0 (*Guinness*), however many utilities have been deprecated since it's release. As a result the default policy used as it stands will produce over 75 file system errors— produced because the files are missing. (*Appendix E - details the files which will produce errors.*)

Full details of producing tripwire policies can be found in the *Tripwire man pages*— an overview of the important details can be found in Appendix B. The tripwire policy does not need to be perfect at first— it can be refined over time. Once the Tripwire policy has been created— it should be placed in the default location:

```
/etc/tripwire/twpol.txt
```

8.3 Initialise the Database

After the post-installation and policy creation stages have taken place, it is time to create the initial database. The creation of the database is a CPU and Disk intensive activity— *rather than anxiously watching for the database to be created you would spend the time better getting a coffee— after you have entered your site passphrase.* The database is configured with the command:

```
/usr/sbin/tripwire --init
```

The database by default will be stored at³:

```
/var/lib/tripwire/zero.dawn2dusk.mellon-collie.net.twd
```

8.4 Checking the Filesystem

The purpose of Tripwire is to check the file system integrity— so it would be useful to know the command to check the file-system. The filesystem can be checked without the need to enter passphrases, this makes it possible to run un-attended:

```
/usr/sbin/tripwire --check
```

A sample Tripwire report is included in Appendix C, highlighting changes after updating the NVidia driver, shadow-utils package, and GnomeMeeting. Of course, the careful system administrator would verify that the offending files highlighted by Tripwire were genuine. The following command will query the files which are part of an rpm package:

```
rpm --query --list Package.rpm
```

8.5 Updating Files

At the time of writing there are **293 security errata, bug-fixes and enhancements** to RedHat 7.3 (Valhalla)— less than a year since it was released. A default install of Valhalla is vulnerable to the Apache chunk encoding flaw [Redhat Network a, 2002]— and the mod_ssl flaws [Redhat Network b, 2002]— the vulnerabilities that gave rise to

³where zero.dawn2dusk.mellon-collie.net is the FQDN domain name of the machine

the **Slapper Worm**. It is the responsibility of the system administrator to ensure that the system has the correct patches applied⁴. By updating the vulnerable Apache we would replace the *httpd* binary in the */usr/sbin* directory (as well as a number of other files). To update the Tripwire database the following command should be run:

```
/usr/sbin/tripwire --update --twrfile \  
/var/lib/tripwire/report/zero.dawn2dusk.mellon-collie.net-20030226-192327.twr
```

Once this command is executed— the default text editor will be launched (*such as vim— the following document explains the intricate user interface of vi— http://www.cisco.com/en/US/products/hw/switches/ps1893/prod_release_note09186a008007cbee.html*). The text editor will be launched with an overview of the changes (*An example of the report can be found in Appendix D*).

Tripwire will highlight the changes it wishes to make, with pre-filled boxes. To authorise a change to the database simply leave the *x* in the box— or remove the *x* if the change should not be entered into the database.

Once the file has been saved and the *vi* editor closes— you will need to supply the local passphrase so that changes may be made to the encrypted database.

8.6 Updating the Policy

The crux of maintaining system security is to not become complacent in the methods used to protect our server. Since an attacker has free access to the Tripwire program— it is easy for them to work around the default policy (*e.g. install a root-kit in a section of the filesystem not scanned.*) It is not unreasonable to expect that the system will upgrade services, for instance upgrading from the Apache 1.3 series to Apache 2.0. It is critical that we are able to update and refine the policy which Tripwire uses.

The first stage is to edit the plain-text policy located in */etc/tripwire/twpol.txt* to reflect the changes that we wish to make to our Tripwire policy. After updating the policy we must execute Tripwire so that it may re-create the signed policy and database:

```
/usr/sbin/tripwire --update-policy /etc/tripwire/twpol.txt
```

On multi-user servers it may be useful to change the security mode to low. In the case of the high security mode (*default*) any changes which would cause the File System Integrity check to fail— will also cause the policy update to fail. If during re-creating your policy changes occur, questions ought to be asked about how suitable the policy is for the system.

```
/usr/sbin/tripwire --update-policy --secure-mode low \  
/etc/tripwire/twpol.txt
```

During the process of updating the Tripwire policy both the local (*used for signing the database*) and the site (*used for signing the policy*) will be required. After parsing the new policy, and checking the file system for integrity— Tripwire will create a new signed database and policy file.

⁴The RedHat Network <https://rhn.redhat.com> provides an easy method of installing and warning about updates as required - a red exclamation mark indicates when an update is required.

8.7 Scheduling Regular Checks

Installation and usage of Tripwire has been discussed— but one step missing from the process has been to create a scheduled process to check the file system. Tripwire does not take care of this itself— instead it hands off to the perfectly able Cron tool⁵. A suitable command to e-mail a Tripwire report (*at the frequency defined in the Crontab.*) would simply be:

```
/usr/sbin/tripwire --check
```

8.8 Protecting Web Content

So far this report has focused upon protecting system binaries— however to bring in a real-world example: The **Recording Industry Association of America** can testify that it is important to ensure that web-pages are not *defaced*. The RIAA's website <http://www.riaa.org> was defaced four times over a few months [The Register, 2003], with them often taking days to fix the problem.

```
RIAA - Own3d by.... ;p
ooh riaa want's to hack Filesharing Users / Servers ? - better lern to secure your
own server... Sorry Admin - had to deactivate ur accounts - they'll be reactivated
after 2 hours
```

```
greetz : Rage_X, BRAiNBUG, SyzL0rd, BSJ, PsychoD + all the others who want to stay
anonymous :] wanna contact ? mailto:h4x0r0815@mail.ru
```

Of course the previous block of text requires referencing to it's author [h4x0r0815@mail.ru, 2003].

Through reading this report you should now have a clear understanding that File System Integrity tools like Tripwire could have been of use to the RIAA. There may be very good reasons why people may want to hack the RIAA web-server— **although of course it is illegal in most countries, including the United Kingdom**⁶ The RIAA did not learn from their mistakes— and with a File System Integrity tool installed, the administrators would be able to understand the scale of the attack— and safely return a corporate homepage to service whilst minimising the downtime.

⁵The cron daemon runs in the background of a server and executes tasks as required— to learn more about Cron visit: <http://www.superscripts.com/tutorial/crontab.html>

⁶Russia however have very liberal laws with regards to computer security when contrasted against America and the Digital Millenium Copyright Act [US Copyright Office, 1998].

Part III

References

References

- [Dobbertin H, 1996] The status of MD5 After a Recent Attack - CryptoBytes Summer 1996 <http://www.rsasecurity.com/rsalabs/cryptobytes/>
- [GNU ls manpage, 2002] Description of the ls utility from the GNU manpage for the ls utility </usr/share/man/man1/ls.1.gz>
- [GNU ps manpage, 2002] Description of the ls utility from the GNU manpage for the ps utility </usr/share/man/man1/ps.1.gz>
- [GNU netstat manpage, 2000] Description of the ls utility from the GNU manpage for the netstat utility </usr/share/man/man8/netstat.8.gz>
- [GNU strace manpage, 1999] Description of the ls utility from the GNU manpage for the strace utility </usr/share/man/man1/strace.1.gz>
- [h4x0r0815@mail.ru, 2003] Record Industry Association of America's homepage <http://www.riaa.org/> Last Accessed: 10th January 2003
Archive available: <http://webpages.charter.net/kevogod/riaahacked.gif>
- [HMSO, 1990] Computer Misuse Act 1990
http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_1.htm
- [Redhat Network a, 2002] Updated Apache packages fix chunked encoding issue
<https://rhn.redhat.com/errata/RHSA-2002-103.html>
- [Redhat Network b, 2002] Updated mod_ssl packages
<https://rhn.redhat.com/errata/RHSA-2002-134.html>
- [RFC 1312, 1992] The MD5 Message-Digest Algorithm
<http://www.faqs.org/rfcs/rfc1321.html>
- [RFC 3174, 2001] US Secure Hash Algorithm 1 (SHA1)
<http://www.faqs.org/rfcs/rfc3174.html>
- [The Register, 2003] RIAA defaced -again!
<http://www.theregister.co.uk/content/55/28817.html>
- [twpolicy manpage, 2002] Description of Property Masks from the Tripwire 2.3 manpage for the twpolicy </usr/man/man4/twpolicy>.
- [US Copyright Office, 1998] The Digital Millenium Copyright Act of 1998
<http://www.loc.gov/copyright/legislation/dmca.pdf>

Unless stated otherwise all web-site references were last checked on the 27th February 2003.

Part IV
Appendicies

Appendix A - Post-Installation

```
[allena28@blackswan tripwire-2.3]$ ./install.sh
```

```
Installer program for:  
Tripwire(R) 2.3 Open Source for LINUX
```

```
Copyright (C) 1998-2000 Tripwire (R) Security Systems, Inc. Tripwire (R)  
is a registered trademark of the Purdue Research Foundation and is  
licensed exclusively to Tripwire (R) Security Systems, Inc.
```

```
LICENSE AGREEMENT for Tripwire(R) 2.3 Open Source for LINUX
```

```
Please read the following license agreement. You must accept the  
agreement to continue installing Tripwire.
```

```
Press ENTER to view the License Agreement.
```

```
... [ GPL v2 ] ...
```

```
Using configuration file install.cfg
```

```
Checking for programs specified in install configuration file....
```

```
/usr/lib/sendmail exists. Continuing installation.
```

```
/bin/vi exists. Continuing installation.
```

```
-----  
Verifying existence of binaries...
```

```
./bin/i686-pc-linux_r/siggen found  
./bin/i686-pc-linux_r/tripwire found  
./bin/i686-pc-linux_r/twprint found  
./bin/i686-pc-linux_r/twadmin found
```

```
This program will copy Tripwire files to the following directories:
```

```
    TWBIN: /tripwire/usr/sbin  
    TWMAN: /usr/man  
    TWPOLICY: /tripwire/etc/tripwire  
    TWREPORT: /tripwire/var/lib/tripwire/report  
    TWDB: /tripwire/var/lib/tripwire  
    TWSITEKEYDIR: /tripwire/etc/tripwire  
    TWLOCALKEYDIR: /tripwire/etc/tripwire
```

```
CLOBBER is false.
```

```
Continue with installation? [y/n]
```

The Tripwire site and local passphrases are used to sign a variety of files, such as the configuration, policy, and database files.

Passphrases should be at least 8 characters in length and contain both letters and numbers.

See the Tripwire manual for more information.

Creating key files...

(When selecting a passphrase, keep in mind that good passphrases typically have upper and lower case letters, digits and punctuation marks, and are at least 8 characters in length.)

Enter the site keyfile passphrase:

Generating key (this may take several minutes)...Key generation complete.

(When selecting a passphrase, keep in mind that good passphrases typically have upper and lower case letters, digits and punctuation marks, and are at least 8 characters in length.)

Enter the local keyfile passphrase:

Verify the local keyfile passphrase:

Generating key (this may take several minutes)...Key generation complete.

Generating Tripwire configuration file...

Creating signed configuration file...

Please enter your site passphrase:

Please enter your site passphrase:

Wrote configuration file: /tripwire/etc/tripwire/tw.cfg

A clear-text version of the Tripwire configuration file /tripwire/etc/tripwire/twcfg.txt has been preserved for your inspection. It is recommended that you delete this file manually after you have examined it.

Customizing default policy file...

Creating signed policy file...
Please enter your site passphrase:

Wrote policy file: /tripwire/etc/tripwire/tw.pol

A clear-text version of the Tripwire policy file
/tripwire/etc/tripwire/twpol.txt
has been preserved for your inspection. This implements
a minimal policy, intended only to test essential
Tripwire functionality. You should edit the policy file
to describe your system, and then use twadmin to generate
a new signed copy of the Tripwire policy.

The installation succeeded.

Please refer to /usr/doc/tripwire/Release_Notes
for release information and to the printed user documentation
for further instructions on using Tripwire 2.3 Open Source for LINUX.

Appendix B - Tripwire Policy

The policy below shows sections of a Tripwire default policy. The default policy with Tripwire serves as an excellent starting base— since it has been tried and tested. However, security cannot be universally defined for all systems— as such the default configuration should only serve as a starting point, from which to refine a near-perfect configuration for each particular situation.

```
# Global Variable Definitions                                     # #
#                                                                 # #
# These are defined at install time by the installation script.  # #
# Manually edit these if you are using this file directly and not # #
# installation script itself.                                   # #

@@section GLOBAL
TWROOT="/usr/sbin/";
TWBIN="/usr/sbin/tripwire";
TWPOL="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME="jellybelly.dawn2dusk.mellon-collie.net";
```

The first section of the Tripwire report defines a number of variables, such as paths and the system hostname which are used in other sections of this policy, and within Tripwire.

```
@@section FS
SEC_CRIT      = $(IgnoreNone)-SHa ; # Critical files that cannot change
SEC_SUID      = $(IgnoreNone)-SHa ; # Binaries with the SUID or SGID flags set
SEC_BIN       = $(ReadOnly) ;      # Binaries that should not change
SEC_CONFIG    = $(Dynamic) ;       # Config files that are changed infrequently but a
SEC_LOG       = $(Growing) ;       # Files that grow, but that should never change ow
SEC_INVARIANT = +tpug ;            # Directories that should never change permission
SIG_LOW       = 33 ;               # Non-critical files that are of minimal security
SIG_MED       = 66 ;               # Non-critical files that are of significant secur
SIG_HI        = 100 ;              # Critical files that are significant points of vu
```

Tripwire now defines a number of security ratings, *those lines beginning with **SEC***. The security ratings defined are applied to the files in the policy below.

In addition to defining the security ratings, we need to create a number of severity ratings *those lines beginning with **SIG***. The severity ratings are *not* applied directly to files— but rather applied to a group of files. The default ratings of 33, 66, 100 are percentages of severity— 100% denoting highly critical changes,— whilst 33% may be used for information purposes only. The severity ratings are only used for the reports.

```

# Tripwire Binaries
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI)
)
{
  $(TWBIN)/siggen          -> $(SEC_BIN) ;
  $(TWBIN)/tripwire       -> $(SEC_BIN) ;
  $(TWBIN)/twadmin        -> $(SEC_BIN) ;
  $(TWBIN)/twprint        -> $(SEC_BIN) ;
}

```

The above section of the Tripwire policy, a section is created with the name **Tripwire Binaries**— whenever a file is triggered from this section it is marked with a severity of **SIG_HI**. In this section we are rating the files *siggen*, *tripwire*, *twadmin* and *twprint* as **SEC_BIN**— binaries that should not change.

```

# Commonly accessed directories that should remain static with regards to owner and group
(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
  /          -> $(SEC_INVARIANT) (recurse = 0) ;
  /home     -> $(SEC_INVARIANT) (recurse = 0) ;
  /etc      -> $(SEC_INVARIANT) (recurse = 0) ;
}

```

The following directories */*, */home* and */etc* are scanned, however since the *recurse* option is set to 0 the filesystem information about the directory is entered into the database— but none of the files or sub-directories.

```

# File System and Disk Administration Programs # #
(
  rulename = "File System and Disk Administration Programs",
  severity = $(SIG_HI)
)
{
  /sbin/badblocks          -> $(SEC_CRIT) ;
  /sbin/dosfsck            -> $(SEC_CRIT) ;
  /sbin/e2fsck             -> $(SEC_CRIT) ;
-- files cut --
  /bin/cp                  -> $(SEC_CRIT) ;
  /bin/cpio                -> $(SEC_CRIT) ;
}

```

```

# Critical configuration files # #
(
  rulename = "Critical configuration files",
  severity = $(SIG_HI)
)
{
  /etc/crontab                -> $(SEC_BIN) ;
  /etc/cron.hourly            -> $(SEC_BIN) ;
-- files cut --
  /etc/resolv.conf            -> $(SEC_CONFIG) ;
  /etc/syslog.conf            -> $(SEC_CONFIG) ;
}

# Critical devices # #
(
  rulename = "Critical devices",
  severity = $(SIG_HI),
  recurse = false
)
{
  /dev/kmem                    -> $(Device) ;
  /dev/mem                     -> $(Device) ;
-- files cut --
  /proc/cmdline                -> $(Device) ;
  /proc/misc                   -> $(Device) ;
}

# Rest of critical system binaries
(
  rulename = "OS executables and libraries",
  severity = $(SIG_HI)
)
{
  /bin                          -> $(SEC_BIN) ;
  /lib                          -> $(SEC_BIN) ;
}

```

In the last section we are defining that the contents of the directories */bin* and */lib* should be monitored— in a group named **Critical System Binaries**. Any file which added or removed from the directory will trigger a change on the directory itself. Each time a file is modified in the directory, the directory will also change— since Unix updates the directories last modified timestamp whenever a file in the directory is updated.

The default Tripwire policy is only a canvas for an individual policy— policies may be customised further with the following options [twpolicy manpage, 2002]:

- - Ignore the following properties

- + Record and check the following properties
- a Access timestamp
- b Number of blocks allocated
- c Inode timestamp (create/modify)
- d ID of device on which inode resides
- g File owner's group ID
- i Inode number
- l File is increasing in size (a "growing file")
- m Modification timestamp
- n Number of links (inode reference count)
- p Permissions and file mode bits
- r ID of device pointed to by inode (valid only for device objects)
- s File size
- t File type
- u File owner's user ID
- C CRC-32 hash value
- H Haval hash value
- M MD5 hash value
- S SHA hash value

There are a number of pre-defined categories which can be used within the tripwire policy [twpolicy manpage, 2002]:

- **ReadOnly** is good for files that are widely available but are intended to be read-only. *Value: +pinugtsdbmCM-rlacSH*
- **Dynamic** is good for monitoring user directories and files that tend to be dynamic in behavior. *Value: +pinugtd-srlbamcCMSH*
- The **Growing** variable is intended for files that should only get larger. *Value: +pinugtdl-srbamcCMSH*
- **Device** is good for devices or other files that Tripwire should not attempt to open. *Value: +pugsdr-intlbamcCMSH*
- **IgnoreAll** tracks a file's presence or absence, but doesn't check any other properties. *Value: -pinugtsdrlbamcCMSH*
- **IgnoreNone** turns on all properties and provides a convenient starting point for defining your own property masks. (For example, mymask = \$(IgnoreNone) -ar;) *Value: +pinugtsdrbamcCMSH-l*

Appendix C - Tripwire Report

Tripwire(R) 2.3.0 Integrity Check Report

Report generated by: root
Report created on: Sat Feb 22 23:18:01 2003
Database last updated on: Never

=====
Report Summary:
=====

Host name: farewell.twilight2starlight.mellon-collie.net
Host IP address: 127.0.0.1
Host ID: None
Policy file used: /tripwire/etc/tripwire/tw.pol
Configuration file used: /tripwire/etc/tripwire/tw.cfg
Database file used: /tripwire/var/lib/tripwire/farewell.twilight2starlight.me
Command line used: usr/sbin/tripwire --check --cfgfile etc/tripwire/tw.cfg

=====
Rule Summary:
=====

Section: Unix File System

Rule Name	Severity Level	Added	Removed	Modified
-----	-----	-----	-----	-----
Invariant Directories	66	0	0	0
* Tripwire Data Files	100	1	0	0
Critical devices	100	0	0	0
Tripwire Binaries	100	0	0	0
* User binaries	66	0	0	25
* Libraries	66	4	4	5
File System and Disk Administration Programs	100	0	0	0
Kernel Administration Programs	100	0	0	0
Networking Programs	100	0	0	0
System Administration Programs	100	0	0	0
Hardware and Device Control Programs	100	0	0	0
System Information Programs	100	0	0	0
Application Information Programs	100	0	0	0
Shell Related Programs	100	0	0	0
(/sbin/getkey)	100	0	0	0
Critical Utility Sym-Links	100	0	0	0

Critical system boot files	100	0	0	0
* Critical configuration files	100	0	0	2
* System boot changes	100	0	0	5
OS executables and libraries	100	0	0	0
Security Control	100	0	0	0
Login Scripts	100	0	0	0
Operating System Utilities	100	0	0	0
Shell Binaries	100	0	0	0
* Root config files	100	0	0	1

Total objects scanned: 23928

Total violations found: 47

=====
Object Summary:
=====

Section: Unix File System

Rule Name: Tripwire Data Files (/tripwire/var/lib/tripwire)

Severity Level: 100

Added:

"/tripwire/var/lib/tripwire/farewell.twilight2starlight.mellon-collie.net.twd"

Rule Name: User binaries (/usr/sbin)

Severity Level: 66

Modified:

"/usr/sbin"
"/usr/sbin/adduser"
"/usr/sbin/chpasswd"
"/usr/sbin/groupadd"
"/usr/sbin/groupdel"
"/usr/sbin/groupmod"
"/usr/sbin/grpck"
"/usr/sbin/grpconv"
"/usr/sbin/grpunconv"
"/usr/sbin/newusers"
"/usr/sbin/pwck"
"/usr/sbin/pwconv"
"/usr/sbin/pwunconv"
"/usr/sbin/useradd"

"/usr/sbin/userdel"
"/usr/sbin/usermod"

Rule Name: Libraries (/usr/lib)
Severity Level: 66

Added:

"/usr/lib/libpt.so.1.4"
"/usr/lib/libpt.so.1.4.7"
"/usr/lib/libopenh323.so.1.11"
"/usr/lib/libopenh323.so.1.11.2"

Removed:

"/usr/lib/libopenh323.so.1.9"
"/usr/lib/libopenh323.so.1.9.10"
"/usr/lib/libpt.so.1.3"
"/usr/lib/libpt.so.1.3.11"

Modified:

"/usr/lib"
"/usr/lib/bonobo/servers"
"/usr/lib/bonobo/servers/gnomemeeting.server"
"/usr/lib/libopenh323.so.1"
"/usr/lib/libpt.so.1"

Rule Name: User binaries (/usr/bin)
Severity Level: 66

Modified:

"/usr/bin"
"/usr/bin/chage"
"/usr/bin/faillog"
"/usr/bin/gnomemeeting"
"/usr/bin/gnomemeeting-config-tool"
"/usr/bin/gpasswd"
"/usr/bin/lastlog"
"/usr/bin/sg"
"/usr/bin/vncviewer"

Rule Name: System boot changes (/var/log)
Severity Level: 100

Modified:

"/var/log/gdm/:0.log"
"/var/log/gdm/:0.log.1"
"/var/log/gdm/:0.log.2"
"/var/log/gdm/:0.log.3"
"/var/log/gdm/:0.log.4"

Rule Name: Critical configuration files (/etc/default)
Severity Level: 100

Modified:
"/etc/default"
"/etc/default/useradd"

Rule Name: Root config files (/root)
Severity Level: 100

Modified:
"/root"

=====
Error Report:
=====

No Errors

*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY; for details use --version. This is free software which may be redistributed or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.

Appendix D - Tripwire Database Update

The following Tripwire report was generated after modifying the configuration of an Apache web-server and starting the *httpd* daemon for the first time.

Tripwire(R) 2.3.0 Integrity Check Report

Report generated by: root
Report created on: Sun Feb 23 02:17:48 2003
Database last updated on: Sat Feb 22 23:59:40 2003

Report Summary:

Host name: farewell.twilight2starlight.mellon-collie.net
Host IP address: 127.0.0.1
Host ID: None
Policy file used: /tripwire/etc/tripwire/tw.pol
Configuration file used: /tripwire/etc/tripwire/tw.cfg
Database file used: /tripwire/var/lib/tripwire/farewell.twilight2starlight.me
Command line used: /tripwire/usr/sbin/tripwire --check --cfgfile /tripwire/e

Rule Summary:

Section: Unix File System

Rule Name	Severity Level	Added	Removed	Modified
Invariant Directories	66	0	0	0
Tripwire Data Files	100	0	0	0
Critical devices	100	0	0	0
Tripwire Binaries	100	0	0	0
User binaries	66	0	0	0
Libraries	66	0	0	0
File System and Disk Administraton Programs	100	0	0	0
Kernel Administration Programs	100	0	0	0
Networking Programs	100	0	0	0
System Administration Programs	100	0	0	0
Hardware and Device Control Programs	100	0	0	0

System Information Programs	100	0	0	0
Application Information Programs				
	100	0	0	0
Shell Related Programs	100	0	0	0
(/sbin/getkey)				
Critical Utility Sym-Links	100	0	0	0
Critical system boot files	100	0	0	0
* Critical configuration files	100	0	0	1
* System boot changes	100	7	0	0
OS executables and libraries	100	0	0	0
Security Control	100	0	0	0
Login Scripts	100	0	0	0
Operating System Utilities	100	0	0	0
Shell Binaries	100	0	0	0
* Root config files	100	0	0	2

Total objects scanned: 23936

Total violations found: 10

=====
Object Summary:
=====

Section: Unix File System

Rule Name: System boot changes (/var/log)
Severity Level: 100

Remove the "x" from the adjacent box to prevent updating the database with the new values for this object.

Added:

- [x] "/var/log/httpd/error_log"
- [x] "/var/log/httpd/ssl_error_log"
- [x] "/var/log/httpd/ssl_access_log"
- [x] "/var/log/httpd/ssl_request_log"
- [x] "/var/log/httpd/access_log"

Rule Name: System boot changes (/var/lock/subsys)
Severity Level: 100

Remove the "x" from the adjacent box to prevent updating the database with the new values for this object.

Added:

"/var/lock/subsys/httpd"

Rule Name: System boot changes (/var/run)

Severity Level: 100

Remove the "x" from the adjacent box to prevent updating the database with the new values for this object.

Added:

"/var/run/httpd.pid"

Rule Name: Critical configuration files (/etc/httpd/conf/httpd.conf)

Severity Level: 100

Remove the "x" from the adjacent box to prevent updating the database with the new values for this object.

Modified:

"/etc/httpd/conf/httpd.conf"

Rule Name: Root config files (/root)

Severity Level: 100

Remove the "x" from the adjacent box to prevent updating the database with the new values for this object.

Modified:

"/root"

"/root/.viminfo"

Object Detail:

Section: Unix File System

Rule Name: System boot changes (/var/log)

Severity Level: 100

Added Objects: 5

Added object name: /var/log/httpd/error_log

Property:	Expected	Observed
* Object Type	---	Regular File
* Device Number	---	5640
* Inode Number	---	245540
* Mode	---	-rw-r--r--
* Num Links	---	1
* UID	---	root (0)
* GID	---	root (0)

Added object name: /var/log/httpd/ssl_error_log

-- detail snipped --

Added object name: /var/log/httpd/ssl_access_log

-- detail snipped --

Added object name: /var/log/httpd/ssl_request_log

-- detail snipped --

Added object name: /var/log/httpd/access_log

-- detail snipped --

Rule Name: System boot changes (/var/lock/subsys)

Severity Level: 100

Added Objects: 1

Added object name: /var/lock/subsys/httpd

Property:	Expected	Observed
* Object Type	---	Regular File
* Device Number	---	5640
* Inode Number	---	163341
* Mode	---	-rw-r--r--
* Num Links	---	1
* UID	---	root (0)
* GID	---	root (0)

Rule Name: Critical configuration files (/etc/httpd/conf/httpd.conf)
Severity Level: 100

Modified Objects: 1

Modified object name: /etc/httpd/conf/httpd.conf

Property:	Expected	Observed
-----	-----	-----
Object Type	Regular File	Regular File
Device Number	5640	5640
* Inode Number	1159506	1159521
Mode	-rw-r--r--	-rw-r--r--
Num Links	1	1
UID	root (0)	root (0)
GID	root (0)	root (0)
* Size	34817	34855
* Modify Time	Wed Oct 9 13:04:22 2002	Sun Feb 23 02:17:17 2003
Blocks	72	72
* CRC32	Ah+8BV	B0JeAT
* MD5	CLa9R+g6/GrB1WZitCC5hB	ALIGWaNjr+zTEh2bePfiTk

Rule Name: Root config files (/root)
Severity Level: 100

Modified Objects: 2

Modified object name: /root

Property:	Expected	Observed
-----	-----	-----
Object Type	Directory	Directory
Device Number	5640	5640
File Device Number	0	0
Inode Number	293761	293761
Mode	drwxr-x---	drwxr-x---
Num Links	25	25
UID	root (0)	root (0)
GID	root (0)	root (0)
Size	4096	4096
* Modify Time	Sat Feb 22 23:26:43 2003	Sun Feb 23 02:17:17 2003
* Change Time	Sat Feb 22 23:26:43 2003	Sun Feb 23 02:17:17 2003

Blocks 8 8

Modified object name: /root/.viminfo

Property:	Expected	Observed
-----	-----	-----
Object Type	Regular File	Regular File
Device Number	5640	5640
File Device Number	0	0
* Inode Number	294172	294165
Mode	-rw-----	-rw-----
Num Links	1	1
UID	root (0)	root (0)
GID	root (0)	root (0)
* Size	6131	6149
* Modify Time	Sat Feb 22 22:07:55 2003	Sun Feb 23 02:17:17 2003
* Change Time	Sat Feb 22 22:07:55 2003	Sun Feb 23 02:17:17 2003
Blocks	16	16
* CRC32	BbuWpd	BC10bP
* MD5	Dk0grs0fXPMNjWEfwEt3/b	B7p+pCPqej0wkx9JydDy7c

=====
Error Report:
=====

No Errors

*** End of report ***

Appendix E - Updates to Tripwire Policy

The following list is files which are not part of a default installation of RedHat 7.3— but are part of the default Tripwire Policy.

```
/sbin/accton -> $(SEC_CRIT) ;
/sbin/fsck.minix -> $(SEC_CRIT) ;
/sbin/mkfs.minix -> $(SEC_CRIT) ;
/sbin/mkpv -> $(SEC_CRIT) ;
/sbin/mtx -> $(SEC_CRIT) ;
/sbin/tapeinfo -> $(SEC_CRIT) ;
/sbin/update -> $(SEC_CRIT) ;
/sbin/adjtimex -> $(SEC_CRIT) ;
/sbin/dhcpd -> $(SEC_CRIT) ;
/sbin/getty -> $(SEC_CRIT) ;
/sbin/ipchains -> $(SEC_CRIT) ;
/sbin/ipchains-restore -> $(SEC_CRIT) ;
/sbin/ipchains-save -> $(SEC_CRIT) ;
/sbin/ipfwadm -> $(SEC_CRIT) ;
/sbin/ipx_configure -> $(SEC_CRIT) ;
/sbin/ipx_interface -> $(SEC_CRIT) ;
/sbin/ipx_internal_net -> $(SEC_CRIT) ;
/sbin/rarp -> $(SEC_CRIT) ;
/sbin/uugetty -> $(SEC_CRIT) ;
/sbin/vgetty -> $(SEC_CRIT) ;
/sbin/linuxconf -> $(SEC_CRIT) ;
/sbin/linuxconf-auth -> $(SEC_CRIT) ;
/sbin/remadmin -> $(SEC_CRIT) ;
/sbin/isapnp -> $(SEC_CRIT) ;
/sbin/kbdrate -> $(SEC_CRIT) ;
/sbin/pnpdump -> $(SEC_CRIT) ;
/sbin/pump -> $(SEC_CRIT) ;
/sbin/shapecfg -> $(SEC_CRIT) ;
/sbin/sash -> $(SEC_CRIT) ;
/bin/gawk-3.0.4 -> $(SEC_CRIT) ;
/bin/consolechars -> $(SEC_CRIT) ;
/bin/sfxload -> $(SEC_CRIT) ;
/bin/vimtutor -> $(SEC_CRIT) ;
/bin/zsh-3.0.8 -> $(SEC_CRIT) ;
/sbin/askrunlevel -> $(SEC_CRIT) ;
/sbin/dnsconf -> $(SEC_CRIT) ;
/sbin/fixperm -> $(SEC_CRIT) ;
/sbin/fsconf -> $(SEC_CRIT) ;
/sbin/ipfwadm-wrapper -> $(SEC_CRIT) ;
/sbin/mailconf -> $(SEC_CRIT) ;
/sbin/managerpm -> $(SEC_CRIT) ;
```

```

/sbin/modemconf          -> $(SEC_CRIT) ;
/sbin/mount.ncp          -> $(SEC_CRIT) ;
/sbin/mount.ncpfs       -> $(SEC_CRIT) ;
/sbin/mount.smb          -> $(SEC_CRIT) ;
/sbin/mount.smbfs       -> $(SEC_CRIT) ;
/sbin/netconf            -> $(SEC_CRIT) ;
/sbin/userconf           -> $(SEC_CRIT) ;
/sbin/uucpconf           -> $(SEC_CRIT) ;
/bin/xnmap               -> $(SEC_CRIT) ;
/usr/tmp                 -> $(SEC_INVARIANT) ;
/var/tmp                 -> $(SEC_INVARIANT) ;
/tmp                     -> $(SEC_INVARIANT) ;
/bin/ksh                 -> $(SEC_BIN) ;
/usr/kerberos/bin/rsh    -> $(SEC_SUID) ;
/etc/rc                  -> $(SEC_CONFIG) ;
/etc/rc.bsdnet           -> $(SEC_CONFIG) ;
/etc/rc.dt                -> $(SEC_CONFIG) ;
/etc/rc.net              -> $(SEC_CONFIG) ;
/etc/rc.net.serial       -> $(SEC_CONFIG) ;
/etc/rc.nfs               -> $(SEC_CONFIG) ;
/etc/rc.powerfail        -> $(SEC_CONFIG) ;
/etc/rc.tcpi              -> $(SEC_CONFIG) ;
/etc/trcfmt.Z            -> $(SEC_CONFIG) ;
/var/lock/subsys/lpd     -> $(SEC_CONFIG) ;
/var/lock/subsys/autofs  -> $(SEC_CONFIG) ;
/var/lock/subsys/reconfig -> $(SEC_CONFIG) ;
/var/lock/subsys/ypbind  -> $(SEC_CONFIG) ;
/etc/conf.linuxconf      -> $(SEC_BIN) ;
/usr/sbin/fixrmtab       -> $(SEC_BIN) ;
/etc/gettydefs           -> $(SEC_BIN) ;
/etc/inetd.conf          -> $(SEC_CONFIG) ;

```

Appendix F - Sample UNIX Commands

The following page is a brief description of the output from a number of UNIX tools. These tools would be used in the event that a system administrator suspects any illegitimate activity.

The tools provide useful feedback to a system administrator to exactly what is happening, and who is using the system— however if these tools themselves have been compromised tracking down an intruder is hard work.

ls - listing of directory contents

```
lrwxrwxrwx    1 root    root      4 Oct  9 22:11 vigr -> vipw
-rwxr-xr-x    1 root    root     12451 Aug 30 21:00 vipw
-rwxr-xr-x    1 root    root    55832 Jun 28  2002 visudo
-rwxr-xr-x    1 root    root    94889 Aug 16  2002 vsftpd
-rwxr-xr-x    1 root    root    70661 Sep  6 20:58 warnquota
```

The first column shows the type of file and permissions.

- - denotes a regular file

l - denotes a link

d - denotes a directory

Ownership of the file is listed in columns three and four, we can see the user who owns the file is root, and the group ownership is also root.

The permissions are made up of **r**ead **w**rite and **e**xecute.

Permissions are set for the owner of the file, the group members, and every one else.

The number following the permissions shows how many links there are to a particular file.

The size of the file in bytes is displayed in the fifth column

The last modification time of the file

The last piece of information is the name of the file/directory/link.

who - show who is logged on

```
root    pts/2      Feb 23 20:06 (pc2-grim1-4-cust34.ldst.cable.ntl.com)
allena28 pts/3      Feb 23 20:08 (ivy.shu.ac.uk)
```

The **who** command shows a list of the users current logged into the system, with the terminal they are attached to, the date they logged in, and the remote host. It is easy to see that this is a useful tool for tracking down an intruder.

ps - process list

```

    root 13963    361  0 21:42:06 ?          0:02 /usr/local/sbin/sshd2
  oracle 26882     1  0 19:21:53 ?          0:00 oracleshu92 (LOCAL=NO)
    blea  3469     1  0  Feb 17 ?          0:03 /usr/openwin/bin/perfmeter
aboulton 19623    218  0 22:44:07 ?          0:00 in.ftpd -l in.ftpd
  nobody 18807  10135  0 22:36:57 ?          0:00 /u2/httpd/bin/httpd -k start
    root 29498  28318  0  Feb 20 pts/22    0:00 login
    root 14056     1  0  Feb 19 ?          152:38 /usr/lib/picl/picld -l -r
  nobody 19227  10135  0 22:41:44 ?          0:00 /u2/httpd/bin/httpd -k start
  jigill  3033      1  0  Feb 17 ?          0:00 xterm -name peter_xterm -title peter_xte
peter_xterm -fg white -bg black -
    root 19589    361  0 22:43:15 ?          0:01 /usr/local/sbin/sshd2
  oracle 24657     1  0  Feb 22 ?          0:00 ora_reco_shu92
aboulton 14463  14451  0 21:46:32 pts/20    0:00 -bash
  oracle 24653     1  0  Feb 22 ?          0:25 ora_ckpt_shu92
    root 18295    361  0 22:28:12 ?          0:01 /usr/local/sbin/sshd2
  nobody 19181  10135  0 22:37:32 ?          0:00 /u2/httpd/bin/httpd -k start
    root 14100  14092  0  Feb 21 pts/33    0:00 -sh
    root 14089    218  0  Feb 21 ?          0:00 in.telnetd -s in.telnetd -a
  nobody 19202  10135  0 22:38:22 ?          0:00 /u2/httpd/bin/httpd -k start
    root 17224    361  0 17:48:03 ?          0:02 /usr/local/sbin/sshd2
    root 19674  19596  0 22:45:38 pts/50    0:00 ps -ef
    root 28315    218  0  Feb 20 ?          0:00 in.telnetd -s in.telnetd -a
  oracle 24665     1  0  Feb 22 ?          0:00 ora_d000_shu92

```

The **ps** command shows a report of the currently running processes on the system. In addition to showing each running process it is possible to show the date a process was started, the CPU execution time, the terminal the process was invoked from, the process id, and the owner of the process.

netstat - print network connections

```

143.52.2.65.80      195.92.168.173.61879 16560      0 24624      0 TIME_WAIT
143.52.2.65.50527  143.52.2.10.1984     24820      0 24820      0 TIME_WAIT
143.52.2.65.50528  217.40.237.157.1313 16560      0 24840      0 TIME_WAIT
143.52.2.65.50529  143.52.2.10.1984     24820      0 24820      0 TIME_WAIT
143.52.2.65.80     216.87.111.152.3096 16560      0 24840      0 ESTABLISHED
143.52.2.65.50531  217.40.237.157.1315 16560      0 24840      0 TIME_WAIT
143.52.2.65.80     80.195.39.179.3093  64047      0 24840      0 TIME_WAIT
143.52.2.65.80     62.252.32.4.54196   15048      0 24624      0 ESTABLISHED

```

The **netstat** command is useful to show current network connections on a system. If an intruder is making use of a system, it may be useful to see the connection established— or more importantly where it is established to.

strace - trace system calls and signals

```
uname({sys="Linux", node="zero.dawn2dusk.mellon-collie.net", ...}) = 0
```

```

brk(0) = 0x8089824
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=81669, ...}) = 0
--
write(5, "DummyUser@jellybelly\'s password: ", 31) = 31
read(5, "p", 1) = 1
read(5, "a", 1) = 1
read(5, "s", 1) = 1
read(5, "s", 1) = 1
read(5, "w", 1) = 1
read(5, "o", 1) = 1
read(5, "r", 1) = 1
read(5, "d", 1) = 1
read(5, "\n", 1) = 1
write(5, "\n", 1) = 1

```

The **strace** command is useful to see what is happening with a file. If a binary is suspected to contain malicious code— the **strace** utility can help. By executing **ssh** from the command line, it is possible to send the **strace** output to a file.

```
strace ssh -lDummyUser jellybelly 2>>/strace.log
```

By comparing the output of **strace** with a clean binary, against the binary which is suspected to contain malicious code— it is possible to rule out abnormal activity (*The diff command will be useful*).

It should be fairly obvious looking at the output from **strace**, that the **ssh** client is writing to the terminal **DummyUser@jellybelly's password:** and then reading the users response character by character.

An updated **ssh** client can be built from source, which writes the server, username and password to a document in a public place for the attacker to view at a later date.



Sheffield Hallam University

File System Integrity

An investigation into File System Integrity and the role it plays in Network Systems Security

Authors: Adam Allen, Jean-Michel Besnard (*Group 4H*)

Course: BSc(Hons) Networks and Communications

Date: February 2003

Lecturer: Mr. Almerindo Grazziano