



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at <http://www.giac.org/registration/gcfa>

SANS GIAC Certified Forensic Analyst Practical

Cover Page

Author: Brian Hutson

Version: 1.2

Title: Forensics and Incident Response : Three
Investigations

Summary:

This document provides three different situations that demonstrate the use of forensic skills and or incident response methodologies including those that relate to legal process.

The first section examines a zip file downloaded from the SANS site and examined in an effort to understand the type of binary contained within the archive and the purpose to which it could be used.

The second section covers the steps taken to analyse the contents of a disk from a third party SUN Solaris system believed to have been compromised by an attacker on the Internet.

Finally the third section develops a scenario supplied by SANS to illustrate the laws and codes of conduct that affect an Internet Service Provider employee in dealing with law enforcement officers in Victoria, Australia.

Table of Contents

SANS GIAC Certified Forensic Analyst Practical	1
Cover Page	1
Overview	3
Part 1: Analyse an Unknown Binary	3
1. Static Analysis	3
Compiling Source Code	13
2. Dynamic Analysis	17
Tracking Program Behaviour	19
Footprint	21
Network Test	21
Conclusion	23
Legal Implications	24
Questions	26
Bibliography and References	27
Part 2: Forensic Analysis on a System	29
Synopsis of Case Facts	29
System Description	30
Hardware	30
Image Media	31
Forensic System	32
Data Transfer	32
MD5 results	33
Media Analysis	33
Timeline Analysis	36
Recover Deleted Files	38
String Search	40
Conclusions	42
Appendix1 InfoLeak Attack	43
Systems Affected	43
Description	43
Appendix 2 TSIG Attack	44
Systems Affected	44
Bibliography and References	45
Part 3: Legal Issues of Incident Handling	46
References	50

Overview

The SANS Practical for the GIAC forensic Analyst Certification requires the completion to a satisfactory level of three discrete tasks.

An unknown binary from a compromised system must be analysed and the results and subsequent conclusion as to its purpose and capabilities documented.

A compromised system must be examined, once again the results documented.

A short discussion on legal issues affecting a particular scenario must be provided.

Part 1: Analyse an Unknown Binary

An unknown file seized from a compromised system has been provided by SANS for analysis of its purpose and capabilities.

This file has been downloaded from the SANS Momgate website in the form of a zip file (binary_v1.2.zip) and placed on a machine prepared for the analysis.

This machine is a Compaq Presario . The system is running the RedHat 7.3 implementation of the GNU/Linux operating system. The kernel is 2.4.18-7 . Appropriate patches have been applied.

A separate environment has been created for the analysis all work will be done inside this directory.

The investigation process has been divided into two distinct parts:

1. *Static Analysis*

The first analysis performed is static analysis where the suspect program itself is not run.

The steps involved generally are:

- Determine the type of file involved
- Review the ASCII strings located within the binary file
- Try and obtain the source code generally through the internet
- Review the source code for similarities and for information regarding implementation and the purpose of the binary
- Compile source code

Upon download the zipfile is examined prior to decompression using the zipinfo utility.

Zipinfo is a utility that provides for the checking of the contents of a zip file without the need to extract said contents. There are several options

that can be used but for the purposes of this examination only two option flags were needed.

- -T print the file dates and times in a sortable decimal format.
- -v list zipfile information in a verbose, multipage format.

For clarity the two options were not combined but rather the command was run twice with the output placed in individual txt files.

```
zipinfo -v binary_v1.2.zip
$> zipinfo -v binary_v1.2.zip
Archive: 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
```

There is no file comment.

Central directory entry #2:

```
-----
atd

offset of local header from start of archive:  75 (0000004Bh) bytes
file system or operating system of origin:    MS-DOS, OS/2 or NT
FAT
version of encoding software:                  2.0
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 14:57:54
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:           81B600 hex
MS-DOS file attributes (20 hex):              arc
```

There is no file comment

The verbose option shows the presence of two files
 atd.md5 is the first file and is from a FAT filesystem. The file has a 32bit CRC and is compressed to 38 bytes.uncmpressed it is 39 bytes.
 The file was according to DOS time last modified on the 22/08/02 at 14:58:08. The apparent file type is text. It appears likely that this file is an md5sum of the second file in the zip.
 The second file is atd and again is from a FAT filesystem. The file has a 32bit CRC and is compressed to 7077 bytes.uncmpressed it is 15348 bytes. The file was according to DOS time last modified on the 22/08/02 at 14:57:04.The apparent file type is binary.

```
zipinfo -T binary_v1.2.zip
$> zipinfo -T binary_v1.2.zip
Archive:  binary_v1.2.zip  7309 bytes  2 files
-rw-rw-rw-  2.0 fat    39 t- defN 20020822.145808 atd.md5
-rw-rw-rw-  2.0 fat   15348 b- defN 20020822.145754 atd
2 files, 15387 bytes uncompressed, 7115 bytes compressed:  53.8%
```

The `-T` option contains similar information to the `-v` flag but it is described in a different manner. From this we can see the files had octal 666 permissions when compressed. This would imply that the binary file would need to have its permissions altered to run and therefore not be operational upon extraction. In other words an additional script or executable is needed for the file to run thus lessening the chances of the attack being made through an email.

The fact that the files come from a FAT system mean that very little information can be gained regarding ownership and times of certain events.

The next step is to unzip the file into our analysis directory.

```
$>unzip binary_v1.2.zip
Archive: binary_v1.2.zip
  inflating: atd.md5
  inflating: atd
$>
```

This operation has presented the two files as previewed by `zipinfo`. To confirm the file types the command `file` was run against both files.

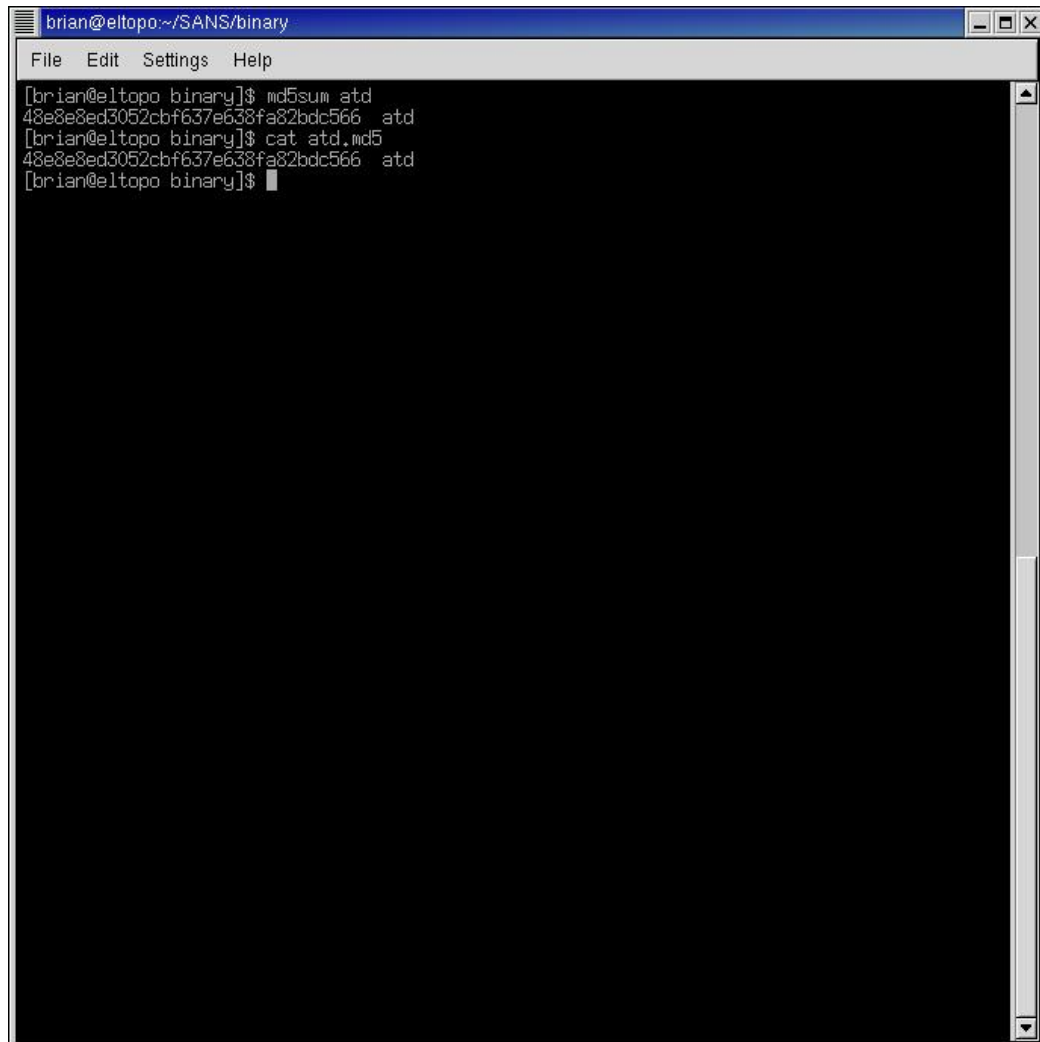
```
$> file atd.md5
atd.md5: ASCII text, with CRLF line terminators
```

This confirms the `zipinfo` report. The file should be able to be read with a text editor so the tool `'less'` is used.

```
48e8e8ed3052cbf637e638fa82bdc566 atd
atd.md5 (END)
```

To verify the assumption that this file is a hash of the binary `atd` `md5sum` is run on the extracted binary `atd`.

```
$> md5sum atd
```

A screenshot of a terminal window titled 'brian@eltopo:~/SANS/binary'. The window has a menu bar with 'File', 'Edit', 'Settings', and 'Help'. The terminal shows the following commands and output:

```
[brian@eltopo binary]$ md5sum atd
48e8e8ed3052cbf637e638fa82bdc566  atd
[brian@eltopo binary]$ cat atd.md5
48e8e8ed3052cbf637e638fa82bdc566  atd
[brian@eltopo binary]$
```

As it is mathematically unlikely that two files would generate the same hash it can be assured that the file extracted is the same file provided at the SANS site.

The atd file itself is now examined using the unix 'file' command. This command attempts to determine the contents of a file using the files header and comparing the information with a list contained in /etc/magic.

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

This output shows the binary is an executable in a format seen in later versions of the Linux operating system. It is compiled for the intel i386 architecture and has been stripped. An executable is stripped when the symbols have been removed from the object files. The atd binary also uses dynamically linked libraries, which means it uses shared system libraries in its operation.

To discover which libraries are needed to run atd the linux tool Readelf was run against the binary with `-d` flag the `d` flag “displays the contents of the file's dynamic section, if it has one”

```
$>readelf -d atd
Dynamic segment at offset 0x3644 contains 17 entries:
  Tag      Type                      Name/Value
0x00000001 (NEEDED)           Shared library: [libc.so.5]
0x0000000c (INIT)             0x8048a70
0x0000000d (FINI)             0x804a8e0
0x00000004 (HASH)            0x80480e8
0x00000005 (STRTAB)           0x80486ac
0x00000006 (SYMTAB)           0x804828c
0x0000000a (STRSZ)            528 (bytes)
0x0000000b (SYMENT)           16 (bytes)
0x00000015 (DEBUG)            0x0
0x00000003 (PLTGOT)           0x804c570
0x00000002 (PLTRELSZ)         400 (bytes)
0x00000014 (PLTREL)           REL
0x00000017 (JMPREL)           0x80488dc
0x00000011 (REL)              0x80488bc
0x00000012 (RELSZ)            32 (bytes)
0x00000013 (RELENT)           8 (bytes)
0x00000000 (NULL)            0x0
```

This shows that the shared library libc.so.5 is needed.

Strings is a utility which prints character strings longer than 4 characters found in a given file. It is a simple yet highly effective way of gaining information about an unknown non-text file. The `-a` flag is used otherwise only the loaded and initialised sections of the binary will be examined.

```
$> strings -a atd
/lib/ld-linux.so.1
libc.so.5
longjmp
strcpy
ioctl
popen
shmctl
geteuid
_DYNAMIC
getprotobynumber
errno
__strtol_internal
usleep
semget
getpid
fgets
shmat
```

_IO_stderr_
perror
getuid
semctl
optarg
socket
__environ
bzero
_init
alarm
__libc_init
environ
fprintf
kill
inet_addr
chdir
shmdt
setsockopt
__fpu_control
shmget
wait
umask
signal
read
strncmp
sendto
bcopy
fork
strdup
getopt
inet_ntoa
getppid
time
gethostbyname
_fini
sprintf
difftime
atexit
_GLOBAL_OFFSET_TABLE_
semop
exit
__setfpucw
open
setsid
close
_errno
_etext
_edata
__bss_start
_end

```

WVS1
f91u
WVS1
pWVS
vuWj
<it    <ut
vudj
<it    <ut
3jTh
j7Wh
Wj7j
Vj7S
j8WS
Vj7S
j8WS
Vj7S
tVj8WS
Vj7S
t'j8WS
jTh8
Wj7j
j7hU
j@hL
@j@hL
jTh8
j      h@
}^j7
}1j7
<WVS
tDWS
lokid: Client database full
DEBUG: stat_client nono
lokid version:      %s
remote interface:   %s
active transport:   %s
active cryptography: %s
server uptime:      %.02f minutes
client ID:          %d
packets written:    %ld
bytes written:      %ld
requests:           %d
N@[fatal] cannot catch SIGALRM
lokid: inactive client <%d> expired from list [%d]
@[fatal] shared mem segment request error
[fatal] semaphore allocation error
[fatal] could not lock memory
[fatal] could not unlock memory
[fatal] shared mem segment detach error
[fatal] cannot destroy shmid
[fatal] cannot destroy semaphore

```

```
[fatal] name lookup failed
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[fatal] Cannot go daemon
[fatal] Cannot create session
/dev/tty
[fatal] cannot detach from controlling terminal
/tmp
[fatal] invalid user identification value
v:p:
Unknown transport
lokid -p (i|u) [ -v (0|1) ]
[fatal] socket allocation error
[fatal] cannot catch SIGUSR1
Cannot set IP_HDRINCL socket option
[fatal] cannot register with atexit(2)
LOKI2 route [(c) 1997 guild corporation worldwide]
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[SUPER fatal] control should NEVER fall here
[fatal] forking error
lokid: server is currently at capacity. Try again later
lokid: Cannot add key
lokid: popen
[non fatal] truncated write
/quit all
lokid: client <%d> requested an all kill
sending L_QUIT: <%d> %s
lokid: clean exit (killed at client request)
[fatal] could not signal process group
/quit
lokid: cannot locate client entry in database
lokid: client <%d> freed from list [%d]
/stat
/swapt
[fatal] could not signal parent
lokid: unsupported or unknown command string
lokid: client <%d> requested a protocol swap
sending protocol update: <%d> %s [%d]
lokid: transport protocol changed to %s
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
01.01
01.01
```

01.01
01.01
01.01
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
.comment
.note

The first thing noticed is the presence of the words loki, lokid and "LOKI2 route 1997 guild corporation" used in the program banner. LOKI (the name comes from the Norse god of trickery) is an example of a covert channel program. That is a program or process that uses a communication path in a manner that was not intended and unauthorised. In the case of Loki an attacker can hide their activity inside a protocol typically ICMP echo request and ICMP echo reply packets. For instance if the perimeter defences of a network allow ICMP traffic the attacker could place a netcat program on a system through the use of Loki to create custom packets that hide the trojan process as ICMP traffic. Communication between the attacker and victim systems is through the ICMP covert channel. LOKI can also utilise DNS queries (typically UDP).

To quote from the PHRACK article by daemon9 aka route

"LOKI2 is an information-tunnelling program. ...In this implementation we tunnel simple shell commands inside of ICMP_ECHO/ICMP ECHOREPLY and DNS namelookup query/reply traffic. To the network protocol analyser this traffic seems like ordinary benign packets of the corresponding protocol. To the correct listener (the LOKI2 daemon)

however, the packets are recognised for what they are. Some of the features offered are: three different cryptography options and on the fly protocol swapping ..."

However at this point we do not have conclusive evidence that the binary is indeed LOKI2 or a part of it (loki2 is a client server program with lokid the listener).

The next step is to attempt to obtain the source code of loki from the internet and any related information on its implementation. Loki's implementation is described in the online magazine Phrack49 and the purported source code is presented in PHRACK 51-06 by daemon9 and alhambra.

Checking the source code one can find the phrases loki, lokid, LOKI2 and the banner can be found.

```
#define L_MSG_BANNER  "\nLOKI2\troute [(c) 1997 guild corporation  
worldwide]\n"
```

This include statement would produce the exact banner found in the binary i.e.

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

Examination of the source code with regard to the strings output of the binary suggests that the binary may well be the listener (lokid) component of LOKI2.

However the code refers to libc.so.6 as opposed to the strings output of libc.so.5 this would imply that this binary if it is indeed loki is an earlier version.

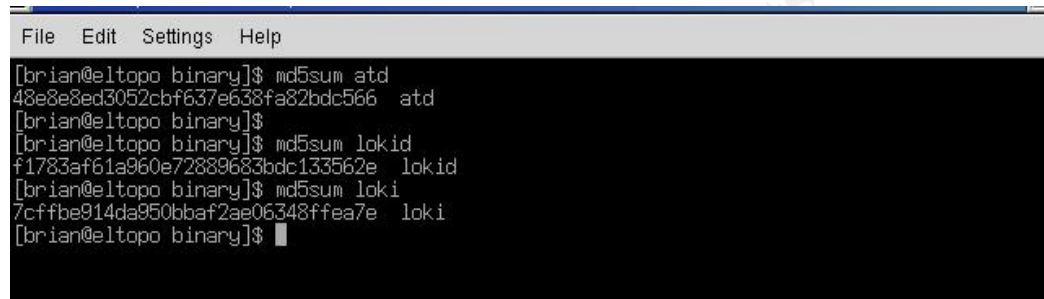
To further correlate the proposition that a version of lokid is indeed the binary further tests need to be undertaken. As this is an unknown binary (despite our suspicions static tests that do not place the network or system at risk should be undertaken first. Compiling the source in order to check for similarities in size and content can be undertaken now at little risk to the environment.

Compiling Source Code

To obtain the relevant source code from the PHRACK article the accompanying extract program is used (this program was tested on another system to ensure its operation was legitimate). At this point the program should be able to be compiled, unfortunately there are several minor errors in the code. These could be a result of sloppy editing or to require more effort from the prospective user. In essence icmp.h and ip.h were in the wrong order and signal.h was wrongly defined.

However once the corrections are made the code can be compiled. Header files from linux 2.0 were the only extra requirement.

Two binaries are compiled the larger being the listener. However neither file is the same size as atd. This could be due to the options chosen in compilation the different libraries used libc.so.6 as opposed to libc.so.5 or even extra usage messages. md5sum confirmed the disparity.



```
File Edit Settings Help
[brian@eltopo binary]$ md5sum atd
48e8e8ed3052cbf637e633fa82bdc566 atd
[brian@eltopo binary]$
[brian@eltopo binary]$ md5sum lokid
f1783af61a960e72889683bdc133562e lokid
[brian@eltopo binary]$ md5sum loki
7cfffbe914da950bbaf2ae06348ffea7e loki
[brian@eltopo binary]$
```

The strings command was used on the new listener binary lokid with the output sent to a file

```
$> strings -a lokid > lokidtmp
```

Then diff was run to compare the lokid output (lokidtmp) with a similar file containing atd's output (atdtmp)

```
$> diff lokidtmp atdtmp
```

```
1,2c1,2
< /lib/ld-linux.so.2
< libc.so.6
---
> /lib/ld-linux.so.1
> libc.so.5
5a6,7
> popen
> shmctl
6a9
> _DYNAMIC
7a11
> errno
13a18
> _IO_stderr_
15a21
> semctl
17a24
> __environ
18a26
> _init
20c28,29
< popen
---
```

```
> __libc_init
> environ
24d32
< __deregister_frame_info
27a36
> __fpu_control
35d43
< __strdup
37a46
> strdup
40a50
> time
41a52
> _fini
44,46c55,56
< stderr
< shmctl
< semctl
---
> atexit
> _GLOBAL_OFFSET_TABLE_
48,49c58,60
< _IO_stdin_used
< __libc_start_main
---
> exit
> __setfpucw
> open
51d61
< __register_frame_info
54,60c64,69
< __cxa_atexit
< __gmon_start__
< GLIBC_2.2
< GLIBC_2.1
< GLIBC_2.1.3
< GLIBC_2.0
< PTRh
---
> _etext
> _edata
> __bss_start
> _end
> WVS1
> f91u
62,75c71,77
< f;49u
< |WVS
< vu^j
< vu`j
```



```
< Pj7j
< j8SW
< Vj7W
< j8SW
< Vj7W
< j8SW
< Vj7W
< j8SW
< Vj7W
< jTh@
---
> pWVS
> vuWj
> <it <ut
> vudj
> <it <ut
> 3jTh
> j7Wh
77,82c79,97
< j7h]
< j@hT
< j@hT
< jTh@
< j    h_
< LWVS1
---
> Vj7S
> j8WS
> Vj7S
> j8WS
> Vj7S
> tVj8WS
> Vj7S
> t'j8WS
> jTh8
> Wj7j
> j7hU
> j@hL
> @j@hL
> jTh8
> j    h@
> }^j7
> }1j7
> <WVS
> tDWS
142a158,165
> 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
151,154c174,175
< 01.01
< 01.01
< 01.01
< 01.01
---
> .symtab
> .strtab
157d177
< .note.ABI-tag
161,163c181
< .gnu.version
< .gnu.version_r
< .rel.dyn
---
> .rel.bss
165a184
> .plt
170,171d188
< .eh_frame
< .dynamic
174a192
> .dynamic

```

As can be seen apart from the expected differences from two compilers used (gcc2.7.1 for atd) (gcc 2.94 for lokid) and libc.so5 used instead of libc.so.6 the two binaries are extremely similar.

Running file on the new binary produces

```

$>file lokid
lokid: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped

```

This is the same as atd.

It is unlikely that this file would be anything other than a version of lokid.

2. *Dynamic Analysis*

The second part of the analysis involves running the suspect binary on a 'safe' system and interprets its interaction with the host operating system. This is done second as there is a risk with this action of damaging the host environment. By performing static analysis first a large amount of information has already been gathered.

The analysis should include:

- Monitoring of time stamps to determine what files are used by the binary.
- Intercept system calls from the binary.
- Monitor the network to determine if any traffic is generated.

Before running the binary some system status details are recorded to assist in analysing the differences made by the suspect program. netstat -anp will show which sockets are open and which processes are responsible for listening to each port.

```
#>netstat -anp >netstat.pre
```

Then lokid is started

```
$>sudo ./lokid
```

LOKI2 route [(c) 1997 guild corporation worldwide]

The following process starts

```
$> ps ax|grep loki |grep -v grep
11441 ?      S    0:00 ./lokid
```

Checking the difference between the original netstat run before starting the process and the netstat -anp run now results in two new entries

```
diff netstat.post ../netstat.pre
```

```
11,12d10
```

```
< raw      0    0 0.0.0.0:1          0.0.0.0:*          7
```

```
11441/lokid
```

```
< raw      0    0 0.0.0.0:255        0.0.0.0:*          7
```

```
11441/lokid
```

This output shows two ICMP sockets are open (the :1 signifies a raw socket of protocol 1, which is ICMP)

The 255 is often seen for sending random data to the target. This output is a symptom of LOKID.

In the interests of safety the lokid process is killed.

ATD then has its permissions changed to allow for execution.

```
$> chmod 755 atd
```

```
$>ls -l atd
```

```
-rwxr-xr-x  1 brian  brian    15348 Aug 22  2002 atd
```

Unfortunately atd does not run on the redhat 7.3 due to the libc library requirement of libc5. Installing these libraries fixes the problem.

atd is then run

```
$>sudo ./atd
```

LOKI2 route [(c) 1997 guild corporation worldwide]

The process table shows

```
$> ps ax|egrep atd|grep -v grep
 847 ?      S    0:00 rpc.statd
2492 ?      S    0:00 ./atd
```

Performing the same check with `netstat` both before and after running the process results in the following:

```
$> diff netstat.preadt netstat.postatd
10a11,12
> raw      0      0 0.0.0.0:1          0.0.0.0:*          7          2492/atd
> raw      0      0 0.0.0.0.0:255        0.0.0.0:*          7
2492/atd
```

This is the same result as obtained from running the lokid binary (the only difference being the process ID).

Tracking Program Behaviour

In order to get an idea of how the program interacts with the operating system the tool strace is used. Strace displays information about file access, network access, memory access and other system calls that an executable makes when run.

The command `strace -f -o atdstrace ./atd` where `-f` allows for the capture of child processes.

[illegible]

From this excerpt line 1 shows the atd binary and the commandline arguments

and is executed with a process ID of 1931.

The next lines are system calls from the operating system to set up the environment for the process to execute in.

The `old_mmap` call (as in line 2) gets the kernel to map a portion of `atd` into memory. This is probably for loading the runtime libraries.

The stat call (as in line 5) obtains information about the file referenced by the descriptor in this case /etc/ld.so.cache. This file is then opened read only. memory is again allocated then the close call releases the file descriptor 3 allowing it to be reassigned.

In line 10 the library libc.so.5 is opened and read

In the next section of interest (lines 23-28) the atd binary obtains the effective userid of the person executing the program. (The owner of the calling process). It then obtains the real userid, and the real group ID of the calling process. In all cases this the 0 or root.

The brk system calls (lines 29 and 30) are used for allocating more memory for the process.

```

39 1931 open("/usr/share/locale.alias", O_RDONLY) = -1 ENOENT (No such file or directory)
40 1931 open("/usr/share/locale/en_AU/LC_MESSAGES", O_RDONLY) = 3
41 1931 fstat(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
42 1931 close(3) = 0
43 1931 open("/usr/share/locale/en_AU/LC_MESSAGES/SYS.LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or directory)
44 1931 open("/usr/share/locale/en/LC_MESSAGES", O_RDONLY) = 3
45 1931 fstat(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
46 1931 close(3) = 0
47 1931 open("/usr/share/locale/en/LC_MESSAGES/SYS.LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or directory)
48 1931 stat("/etc/locale/C/libc.cat", 0xbfffd7e0) = -1 ENOENT (No such file or directory)
49 1931 stat("/usr/share/locale/C/libc.cat", 0xbfffd7e0) = -1 ENOENT (No such file or directory)
50 1931 stat("/usr/share/locale/libc/C", 0xbfffd7e0) = -1 ENOENT (No such file or directory)
51 1931 stat("/usr/share/locale/C/libc.cat", 0xbfffd7e0) = -1 ENOENT (No such file or directory)
52 1931 stat("/usr/local/share/locale/C/libc.cat", 0xbfffd7e0) = -1 ENOENT (No such file or directory)
53 1931 socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
54 1931 sigaction(SIGUSR1, {0x804a6b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x4004def8) = 0
55 1931 socket(PF_INET, SOCK_RAW, IPPROTO_RAW) = 4
56 1931 setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
57 1931 getpid() = 1931
58 1931 getpid() = 1931
59 1931 shmget(2173, 240, IPC_CREAT|0) = 16318476
60 1931 shmget(2355, 1, IPC_CREAT|0x180|0600) = 65538
61 1931 shmat(16318476, 0, 0) = 0x40008000
62 1931 write(2, "\nLOKID:troute [(c) 1997 guild cor"... 52) = 52
63 1931 time([1047220753]) = 1047220753
64 1931 close(0) = 0
65 1931 sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x4004def8) = 0
66 1931 sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x4004def8) = 0
67 1931 sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x4007eef8) = 0
68 1931 fork() = 1932
69 1931 close(4) = 0
70 1931 close(3) = 0
71 1931 semop(65538, 0xbfffd64, 2) = 0
72 1931 shmdt(0x40008000) = 0
73 1931 semop(65538, 0xbfffd64, 1) = 0
74 1931 _exit(0) = ?
75 1932 setsid() = 1932
76 1932 open("/dev/tty", O_RDWR) = -1 ENXIO (No such device or address)
77 1932 chdir("/tmp") = 0
78 1932 umask(0) = 022
79 1932 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x4004def8) = 0
80 1932 alarm(3600) = 0
81 1932 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x4007eef8) = 0
82 1932 read(3, 0x804c78c, 84) = ? ERESTARTSYS (To be restarted)
83 1932 --- SIGTERM (Terminated) ---

```

Basic environment variables are now set for the locale (line 39-44) then an fstat call obtains information about the file permissions and size.

According to the PHRACK articles Lokid has to perform its own TCP/IP processing therefore if atd is a version of lokid then it should do the same. Lines 53 and 55 do indeed reveal this as raw sockets are created. In line 57 and 58 getpid checks on the process ID (1931)

Now a banner is presented (line 62) which is the one presented on running the binary.

Time is checked (line 63)

atd then forks a new process (1932) and then exits the parent (line 74).

The new process attempts to open a tty port (line 76) this action fails but it does change directory to /tmp (line 77). One of the known signatures of loki is that it places the user in the /tmp directory

Finally the atd process 1932 is killed manually (line 83).

Footprint

To gain further understanding of the 'footprint this binary has on the system the mactime tool from the coroners toolkit can be used.

The atd binary was run for approximately a minute before being killed with the starting and finishing times recorded. At this point mactime was run to collate all activity in the last twenty four hours.

```
$> tct-1.09/bin/mactime -R -d / "03/09/2003" >atdmactime
```

Then the time period in which atd was run was examined

```
7398 4096 .a. drwxr-xr-x brian brian /home/brian/SANS/tct-1.09/wan/wan5
7399 Mar 10 03 04:36:29 2176218 .a. -rwxr-xr-x root root /usr/1486-linux-libc5/lib/libc.so.5.4.44
7400 25386 .a. -rwxr-xr-x root root /lib/ld-linux.so.1.9.5
7401 15348 .a. -rwxr-xr-x brian brian /home/brian/SANS/binary/atd
7402 Mar 10 03 04:36:35 48736 .a. -rwxr-xr-x root root /lib/libproc.so.2.0.7
7403 114076 .a. -rwxr-xr-x root root /bin/grep
7404 63304 .a. -r-xr-xr-x root root /bin/ps
7405 Mar 10 03 04:36:53 2862 .a. -rw-r--r-- root root /etc/security/pam_env.conf
7406 278 .a. -rw-r--r-- root root /etc/pam.d/sudo
7407 71016 .a. -rwxr-xr-x root root /usr/lib/libc.so.2.7
7408 50425 .a. -rwxr-xr-x root root /lib/security/pam_unix.so
7409 14617 .a. -rwxr-xr-x root root /lib/security/pam_limits.so
7410 12989 .a. -rwxr-xr-x root root /lib/security/pam_env.so
7411 84680 .a. ---s---x--x root root /usr/bin/sudo
7412 35698 .a. -rwxr-xr-x root root /lib/libpam.so.0.75
7413 643 .a. -rw-r--r-- root root /etc/pam.d/system-auth
7414 4910 .a. -rwxr-xr-x root root /lib/security/pam_deny.so
7415 579 .a. -r--r--r-- root root /etc/sudoers
7416 951 .a. -r----- root root /etc/shadow
7417 6594 m,c -rw----- root root /var/log/secure
7418 12471 .a. -rwxr-xr-x root root /lib/security/pam_stack.so
7419 210 .a. -rw-r--r-- root root /etc/pam.d/other
7420 14540 .a. -rwxr-xr-x root root /lib/security/pam_cracklib.so
7421 7764 .a. -rwxr-xr-x root root /bin/kill
7422 Mar 10 03 04:37:28 32768 .a. drwxr-xr-x root root /usr/bin
7423 Mar 10 03 04:46:18 3703 .a. -rw-r--r-- root root /usr/share/groff/1.17.2/font/devlatin1/B
```

This output shows libc.so.5 and ld-linux.so.1.9.5 being accessed at the same time (04:36:29) as atd itself.

Then libproc.so.2.0.7 was accessed this is part of the procs package.

Then several process are run that relate to the finding of the process ID and use of the sudo privilege command which is needed to kill the process which occurs at 04:37:28

Network Test

A further test can be carried out by placing a second machine in the test network and using the compiled loki client to talk to the atd binary which it is supposed is lokid. The second machine is running the debian variant of GNU/Linux.

By running tcpdump on our redhat system with the atd daemon listening the loki client is used to setup a covert channel

```

tcpdump: listening on eth0
12:28:29.059372 192.168.1.101 > 192.168.1.100: icmp: echo request (ttl 64, id 16093, len 84)
    4500 0054 3edd 0000 4001 b7b2 c0a8 0165
    c0a8 0164 0800 02c7 c928 01f0 b115 790a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:29.059532 192.168.1.100 > 192.168.1.101: icmp: echo reply (ttl 64, id 57273, len 84)
    4500 0054 dfb9 0000 4001 16d6 c0a8 0164
    c0a8 0165 0000 0ac7 c928 01f0 b115 790a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:29.088412 192.168.1.100 > 192.168.1.101: icmp: echo reply (ttl 64, id 57274, len 84)
    4500 0054 dfba 0000 4001 16d5 c0a8 0164
    c0a8 0165 0000 cac6 c928 01f0 f115 790a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:30.951527 192.168.1.101 > 192.168.1.100: icmp: echo request (ttl 64, id 16094, len 84)
    4500 0054 3ede 0000 4001 b7b1 c0a8 0165
    c0a8 0164 0800 0dcf c928 01f0 b10d 6e0a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:30.951697 192.168.1.100 > 192.168.1.101: icmp: echo reply (ttl 64, id 57275, len 84)
    4500 0054 dfbb 0000 4001 16d4 c0a8 0164
    c0a8 0165 0000 15cf c928 01f0 b10d 6e0a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:30.978199 192.168.1.100 > 192.168.1.101: icmp: echo reply (ttl 64, id 57276, len 84)
    4500 0054 dfbc 0000 4001 16d3 c0a8 0164
    c0a8 0165 0000 d5ce c928 01f0 f10d 6e0a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:32.227103 192.168.1.101 > 192.168.1.100: icmp: echo request (ttl 64, id 16095, len 84)
    4500 0054 3edf 0000 4001 b7b0 c0a8 0165
    c0a8 0164 0800 02c7 c928 01f0 b115 790a
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000
12:28:32.227265 192.168.1.100 > 192.168.1.101: icmp: echo reply (ttl 64, id 57277, len 84)

```

This is the output from the client connecting to the server using the command
 %> ./loki -d 192.168.1.100

Following the successful connection the current directory was determined (/tmp) the cd command (which does not work as per typical loki behaviour) and the ls command were run on the target system (where atd was running).

```

Unknown transport.
melbhutson:/home/hutbr01/SANS/practical/binanalysis# ./loki -d 192.168.1.100

LOKI2 route [(c) 1997 guild corporation worldwide]
loki> ls
1.0-20030304-fixes2.tar.gz
compat-glibc-6.2-2.1.3.2.i386.rpm
cxoffice.reg0.log
cxoffice.reg1.log
cxoffice.reg3.log
cxoffice.setup.log
libc-5.4.44-1glibc.i386.rpm
orbit-brian
orbit-pam
orbit-root
scrollkeeper-tempfile.0
scrollkeeper-tempfile.1
strace-4.4-4.i386.rpm
tcpdump-3.6.2-12.i386.rpm
loki> cd
loki> pwd
/tmp
loki> cd /
loki> pwd
/tmp
loki>

```

According to “Intrusion Signatures and Analysis” Loki can be identified by its sequence number, which remains constant. From the above output this does indeed appear to be the case with 01f0 as the sequence number in all packets.

As a final confirmation atd is killed and the loki client attempts to connect again to the target host.

```

[brian@eltopo plugtmp]$ ps ax|grep atd|grep -v atd
[brian@eltopo plugtmp]$ ps ax|grep atd|grep -v grep
  847 ?        S          0:00 rpc.statd
 2492 ?        S          0:00 ./atd
[brian@eltopo plugtmp]$ sudo kill 2492
sudo: Can't get working directory!
[brian@eltopo plugtmp]$ ps ax|grep atd|grep -v grep
  847 ?        S          0:00 rpc.statd
[brian@eltopo plugtmp]$

```

executing the ls command on the loki client results in the following:

```

[529]:~/SANS/practical/binanalysis$> sudo ./loki -d 192.168.1.100

LOKI2 route [(c) 1997 guild corporation worldwide]
loki> ls

Alarm.

loki: no response from server (expired timer)
loki>

```

Conclusion

The binary atd is a version of the lokid daemon. While the available source code does not match in terms of one way hash (e.g. md5) this can be

explained in the fact the source code is for a later set of libraries (libc.so.6 as opposed to libc.so.5) and perhaps different options used in compilation. The output of strings and strace suggest great similarities between lokid and atd. Known behaviour of atd such as TCP/IP processing and a netstat output that is the recognised signature of lokid were all noted. In addition a loki daemon was able to connect to the host system through the atd listener. The tcpdump log was again characteristic of loki behaviour. When atd was stopped the loki daemon was unable to connect.

ATD should now be listed as a possible alias to the lokid daemon. The atd executable is present on a large number of GNU/Linux systems and is used for scheduling. This makes the name ideal for hiding a trojan.

Legal Implications

The ATD binary is in reality the listener component of LOKI2. The implication is therefore that the system containing the binary was 'OWNED' by a 'cracker'. It is this person who is referred to with regard to legal implications rather than the user of the system (although there is a possibility they are the same).

The first question to define is whether this trojan was ever run on the system. Unfortunately because the binary in the zipfile did not have its execute permissions set it is impossible to ascertain the capability or otherwise of the atd trojan.

Even assuming the binary was present on the system in an 'inactive' state internal company policies would be violated. These would include the download and storage of known 'hacking' tools on a system.

The storage of the file on the system would also fall under what the Internet Activities Board consider unethical and unacceptable behaviour. ("Wasting Resources through purposeful actions").

If the file were indeed executed then company policy covering loading and use of non-standard software would be affected.

In addition there are company policies covering the known disruption to the network and related services. As loki covert channel is not used in any normal network related task any policies relating to misuse of system and network equipment both on the company network and any other network accessible from the company can be cited.

The Internet Activities Board's acts of unacceptable behaviour lists disrupting the internal use of the Internet if this product was used across the Internet then this too would be breached.

Assuming the incident happened in Victoria, Australia then several laws both Federal (Criminal Code Act 1995) (CyberCrime Act 2001) and Victorian State (Property and Computer Offences Act 2003, Crimes Act 1958, Summary Offences Act 1966) will have been broken.

The Property Damage and Computer Offences Act 2003 amends the Crimes Act 1958 to create offences relating to computers.

The sections from which indictable offences could be invoked are:

- (247B) Unauthorised access, modification or impairment with intent to commit serious offence.
 - (1) A person who causes any unauthorised computer function—
 - (a) knowing it is unauthorised; and
 - (b) with the intention of committing a serious offence

This section would be affected if malicious intent could be proved and if it was discovered that the binary had been run that act being an unauthorised computer function
- (247C) Unauthorised modification of data to cause impairment
- (247D) Unauthorised impairment of electronic communication
- (247E) Possession of data with intent to commit serious computer offence.

This section would be affected if malicious intent could be proved. The owner or user of the system on which the binary was found could be assumed liable.
- (247F) Producing, supplying or obtaining data with intent to commit serious computer offence.

This section would be affected by the mere presence of the binary on the system even in compressed form provided malicious intent could be proved.
- (247H) Unauthorised impairment of data held in computer disk, credit card or other device

This section would be affected if there is impairment to the reliability, security or operation of data which execution of the binary would cause, or if that intention can be proved to exist

Section 9A of the Victorian Summary Offences Act 1966, which states
A person may not “gain access or enter a computer system or part of a computer system without authority”

This section covers not only entry by external parties but also employees acting outside their permitted tasks.

Precedence for this was delivered in following appeal judgement

Director of Public Prosecutions v Murdoch [1993] 1 VR 406, Hayne J

The file owner would also be in violation of several elements of the Federal ‘Model Criminal Code (Implemented in the Cybercrime Bill s477.1 -- 478.4 of the Criminal Code Act) which is a complementary scheme to support existing state legislation.

The indictable offences that could be invoked are:

- (477.3) Unauthorised impairment of electronic communications
- (478.1) Unauthorised access, modification or impairment of electronic communication where the person is not entitled to cause that access.

The lesser summary offences include:

- Unauthorised access, modification or impairment of data held in a computer disk.
- Unauthorised access or modification of restricted data
- Possession of data (in this case the trojan) with intent to commit a computer offence.

Questions

The type and nature of the questions asked a suspect in this case would in large part depend on the nature of the information already gathered about the person (e.g. repeat offender, contractor, disgruntled employee, threats made by suspect, apparent motive, opportunity and means). In addition the type of person being interviewed and their position in the company would also have a bearing on the questions asked. The place of interview and presentation of evidence and questions can have a considerable bearing on the success of any interview.

The following questions can be viewed as guidelines for real questions posed in an interview:

1. Are you aware of any unusual behaviour on the network.
-----this may illicit comment on the ICMP traffic seen
2. Do you know of any new software on the <compromised> system
-----this gives the suspect a chance to mention the binary
3. What do you think of Linux.? what is it like to program in?
-----as the binary was compiled on linux this may give a clue
4. You know the network pretty well...do you know any backdoors to the systems if there is a password lost?
-----the chance to demonstrate superior knowledge is very tempting
5. What do you think of the security of the network/systems do you know of any way to show it is lax?
-----the chance to demonstrate superior knowledge is very tempting
6. Do you know of software that can get round our firewalls?

-----the chance to demonstrate superior knowledge is very tempting

7. Are you aware of the Acceptable Use Policy of the Company.?
8. What is wrong with <target user>'s computer he/she is always complaining?

-----this plays on any resentment that may exist in the suspect to the user of the compromised system (this could relate to a group if more appropriate)

9. How much network programming have you done?

-----this may reveal the skills possessed by the programmer

Bibliography and References

Daemon9. "Project Loki." File 6 of 16 Issue49 Volume 7 PHRACK Magazine August 1996

<http://www.phrack-don't-give-a-shit-about-dcma.org/phrack/49/P49-06>

Daemon9. "LOKI2 (the implementation)." Article 6 of 17 Issue51 Volume 7 PHRACK Magazine 7th September 1997

<http://www.phrack-don't-give-a-shit-about-dcma.org/phrack/51/51-06>

Various Linux Man Pages

Nortcutt, Cooper, Fearnow, Fredrick, pp246-250 "Intrusion Signatures and Analysis" New Riders Publishing 2001

http://www.iss.net/security_center/static/1452.php

W. Richard Stevens "Advanced Programming in the UNIX Environment" Addison-Wesley 1993

W. Richard Stevens pp69-89 "TCP/IP Illustrated, Volume 1" Addison-Wesley 1994

Ed Skoudis pp465-466 "Counter Hack A Step by Step Guide to Computer Attacks and Effective Defenses" Prentice Hall 2001

MODEL CRIMINAL CODE

<http://www.efa.org.au/Issues/Security/>

<http://www.law.gov.au/>

<http://www.liv.asn.au/research/library/cat/7726.html>

Model Criminal Code Discussion Paper January 2000

Chapter 4 Damage and Computer Offences pp 139-141 "Computer Trespass in Victoria: The Meaning of Unauthorised Entry"

CyberCrime Act

<http://www.aph.gov.au/library/pubs/bd/2001-02/02bd048.htm>

<http://nationalecurity.ag.gov.au/www/nationalecurityhome.nsf/HeadingPagesDisplay/Legislation+NS?OpenDocument>

Telecommunications Act 1997

<http://scaletext.law.gov.au/html/comact/11/6501/top.htm>

<http://www.efa.org.au/Issues/Privacy/surveillance.html>

Crimes (Property Damage and Computer Offences) Act 2003

Crimes (Property Damage and Computer Offences) – Introduction Print

<http://www.dms.dpc.vic.gov.au/pdocs/bills/B01382/B01382I.html>

Internet Activities Board Statement of Policy

<http://www.iab.org/iab/>

<http://catless.ncl.ac.uk/Risks/8.08.html>

© SANS Institute 2003, Author retains full rights.

Part 2: Forensic Analysis on a System

Synopsis of Case Facts

The Security Officer of ACME Company, (not real name), contacted me with a request to provide a forensic analysis of a host, which he believed may well have been compromised a couple of days earlier. The system had been disconnected from the network and placed on a separate VLAN. I had the system connected to an unused hub until I reached the site.

At the site I was given a copy of the Incident Report. In my presence the local system administrator caused the system to halt. This action occurred inadvertently while attempting to provide me with access to the terminal to run commands (netstat, ps, w) that would assist in capturing the present state of the system.

Faced with a situation in which I could not gain any live data I removed the disk and took it offsite to copy the data to the analysis system.

The actions leading to the call were detailed in the Incident report supplied to me:

On the 15th March 2003, the ACME companies Intrusion Detection Systems logged a port scan of the public demilitarised zone looking for DNS services (e.g. nmap -SU -p 53 xxx.xxx.xxx.0/24). The Scan resulted in the server in question, (which shall be called for the purposes of this document DNS_1), being identified.

At 0615 the Intrusion Detection Systems raised alerts over the same attacker attacking DNS_1 with an INFOLeak query. (see Appendix1) This attack consists of sending a crafted inverse query to BIND. The response from the DNS server will contain portions of the STACK memory used by the BIND process. This reveals environment variables and memory layout and often precedes a TSIG attack.

As noted this attack was followed with an alert for the TSIG attack on DNS_1. This attack is a buffer overflow attack taking advantage of parsing of Transaction Signature Requests.

(Both of these alerts are triggered by an entry in the IDS signature file. This is by no means foolproof and there is the possibility of false positive entries for these attacks)

At 0630 Administration staff were alerted of the incident but took three hours to come on site in this time the system was left running. Upon arrival on the site the administrators accessed the system and made the determination that the system had been compromised through the detection of non-standard services installed and running (apache, telnet, FTP). At this point a backup of the server was attempted. At 1110 the firewall protecting DNS_1 was updated to block external DNS queries. A mail was sent to the abuse address ISP of the suspected attacker at 1400. At 1500 the backup was found to have failed due to the new firewall ruleset blocking access to the backup server. Rules were amended to open up internal access to and from the compromised server. At 1506 the system was disconnected from the network.

At 1720 the incident response team left the area.

Whilst it is not within the scope of this document to critique the actions of ACME Companies Incident Response Team it must be noted that the actions of the administrators may well have severely damaged the evidence. A better method would have been to remove the system immediately from the network. Halt the system without a sync. Remove one of the two disks (the system is mirrored using SUN disksuite) reboot the system and do an investigation on the disk still in the system. This would ensure a pristine copy of the compromised system would be kept intact.

System Description

The system involved in the compromise DNS_1 is a SUN SPARC 220R. The system was running Solaris 8 (SUNOS5.8) as its operating system. The serial number of the unit was SN xxxxxxxxx. DNS_1 was used as ACME companies primary DNS server for external and internal queries. It was located in a secured Computer room that was housed inside ACME companys' offices. The system contained two disks, which were mirrored (RAID 1) using the SUN Disksuite product. Given the time the system was left up following the purported attack the disks would have synced the data between them.

Hardware

The item obtained was one of the two disks present in the system.

Date 15 March 2003	Acme I.T	Incident Number 20030315	
System Owner XXXXXXX	Signature of Owner XXXXXXX	Item Tag Number 79393	
Description of Item Fujitsu Disk Model MAG309ILC ID:JW ULTRA2 SCSI Part Number CA01776-B32300SU Serial Number 40145569 0066152-9949445569 Capacity 8.43 Gb.			
Recipient of Evidence Brian Hutson		Position XXXXX	Signature XXXXXX
Chain of Custody			
From: ACME Company Location xxxx street XXXX Victoria	Date: 170303	Reason: Forensic Analysis of disk	To: Forensic Company Location:

			xxxx street XXXX Victoria
--	--	--	---------------------------------

Image Media

The system disk was placed inside a SUN Sparc 10 running the Solaris8 operating system. This procedure was done while the system was up to prevent against the possibility of the system booting off the evidence media. Solaris8 is capable of dynamically adding devices to the system. This is achieved through the following commands.

```
# > drvconfig
```

```
#> disks
```

The format command can then be used to see if the incident disk (Fujitsu 8.43gb) has been detected

```
#> format
```

```
disk0 c0t0d0 Fujitsu 4.79GB
```

```
disk1 c0t1d0 Fujitsu 8.43GB
```

```
format> quit
```

An md5sum of the entire disk is taken (slice2 represents the whole disk in Solaris)

```
# > md5sum /dev/dsk/c0t1d0s2
0f614d0f2aec9a2f23a4a7392bb49b5f /dev/dsk/c0t1d0s2
~
~
~
```

The first slice of the disk is now mounted in a read-only state so as to gain partition and mount point information. This slice is chosen because it is typically the root partition containing the relevant information.

```
#>mount -F ufs -r -o noatime,nosuid /dev/dsk/c0t1d0s0 /mnt
```

A check of the etc/vfstab file for the disk shows the following

```
#device      device      mount      FS      fsck      mount      mount
#to mount    to fsck     point      type     pass     at boot   options
#
fd           -           /dev/fd    fd        -         no        -
/proc        -           /proc      proc      -         no        -
/dev/md/dsk/d0 /dev/md/rdisk/d0 /          ufs       1         no        -
/dev/md/dsk/d10 -           -          swap      -         no        -
/dev/md/dsk/d30 /dev/md/rdisk/d30 /usr       ufs       1         no        -
/dev/md/dsk/d40 /dev/md/rdisk/d40 /var       ufs       1         no        -
/dev/md/dsk/d50 /dev/md/rdisk/d50 /opt       ufs       2         yes       -
swap         -           /tmp       tmpfs     -         yes       -
#/dev/dsk/c0t6d0s0 /dev/rdisk/c0t6d0s0 /cdrom     hsfs     - no ro
```

The disk is then unmounted.

At this point we can copy the data to the forensic system as images of each partition and in addition know what the images are from (e.g. /usr, /var).

An md5sum is taken of each partition before any data is moved.


```
#-----md5sum on compromised disk-----#
#--/usr/local/bin/md5 run on santesagre
#MD5 (/dev/dsk/c0t1d0s2) = 0f614d0f2aec9a2f23a4a7392bb49b5f
#---/usr/local/bin/md5sum run on santesagre
#0f614d0f2aec9a2f23a4a7392bb49b5f /dev/dsk/c0t1d0s2
#-----/usr/local/bin/md5sum run on santesagre slice 0 --#
# md5sum /dev/dsk/c0t1d0s0
#1777bce42b21b70ea9885313d2f5f431 /dev/dsk/c0t1d0s0
#-----/usr/local/bin/md5sum run on santesagre slice 1 --#
# md5sum /dev/dsk/c0t1d0s1
#2f50ef10a4bd36f7c3cca63f2be3fb31 /dev/dsk/c0t1d0s1
#-----/usr/local/bin/md5sum run on santesagre slice 3 --#
# md5sum /dev/dsk/c0t1d0s3
#32cec9b9a21620d421d19a0bed4fef64 /dev/dsk/c0t1d0s3
#-----/usr/local/bin/md5sum run on santesagre slice 4 --#
# md5sum /dev/dsk/c0t1d0s4
#987637b54899101a666239ed23282ff4 /dev/dsk/c0t1d0s4
#-----/usr/local/bin/md5sum run on santesagre slice 5 --#
# md5sum /dev/dsk/c0t1d0s5
#4f4b7ab62f99d95cbb67087d56936ac7 /dev/dsk/c0t1d0s5
#-----/usr/local/bin/md5sum not run on santesagre slice 6 slice zero size--#
#-----/usr/local/bin/md5sum run on santesagre slice 7 --#
# md5sum /dev/dsk/c0t1d0s7
#fc5e9e53bcd522a8c4b8e365846bf937 /dev/dsk/c0t1d0s7
#####
```

Forensic System

The Forensic system is a Toshiba Tecra 8100 laptop running the Debian version of GNU/Linux. Bearing in mind the size of the DNS_1 disk a 10gb disk is placed in the system in addition to the original system disk. This disk will be used to hold the images of the DNS_1 filesystems. A single partition is created for the entire disk.

The disk is 'wiped' of old data using the dd command.

```
#> dd if=/dev/null of=/dev/hdc1
```

```
#> dd if=/dev/random of=/dev/hdc1
```

```
#> dd if=/dev/zero of=/dev/hdc1
```

These three commands write nulls, zeros and random characters throughout the disk ensuring there is no old data that could pollute the investigation.

Data Transfer

Netcat is started as a listener service on the forensic system

```
#> nc -l 1234 > /dev/hdc1/slice0.dd
```

On the Sparc10 containing the DNS_1 disk the dd command is run piping into the netcat command, which then transfers the data across the network to the forensic system.

```
#> dd if=/dev/dsk/c0t1d0s0 |nc -p 1234
```

Once the transfer has completed the data is checked through the md5sum program with the output on the forensic system compared against the md5sum taken prior to the transfer.

The netcat listener on the forensic system is then killed and restarted for the next image.

```
#> nc -l -p 1234 > /dev/hdc1/slice3.dd
```

On the Sparc10 containing the DNS_1 disk the dd command is again run piping into the netcat command, which then transfers the data across the network to the forensic system.

```
#> dd if=/dev/dsk/c0t1d0s3 |nc 192.168.1.101 1234
```

This process is repeated for slice1, slice4, slice5 and slice7 each time the md5sum of the transferred image compared against the original.

MD5 results

```
#####
#-----md5sum on compromised disk-----#
#--/usr/local/bin/md5 run on santesagne
#MD5 (/dev/dsk/c0t1d0s2) = 0f614d0f2aec9a2f23a4a7392bb49b5f
#---/usr/local/bin/md5sum run on santesagne
#0f614d0f2aec9a2f23a4a7392bb49b5f /dev/dsk/c0t1d0s2
#-----/usr/local/bin/md5sum run on santesagne slice 0 --#
# md5sum /dev/dsk/c0t1d0s0
#1777bce42b21b70ea9885313d2f5f431 /dev/dsk/c0t1d0s0
#-----/usr/local/bin/md5sum run on santesagne slice 1 --#
# md5sum /dev/dsk/c0t1d0s1
#2f50ef10a4bd36f7c3cca63f2be3fb31 /dev/dsk/c0t1d0s1
#-----/usr/local/bin/md5sum run on santesagne slice 3 --#
# md5sum /dev/dsk/c0t1d0s3
#32cec9b9a21620d421d19a0bed4fef64 /dev/dsk/c0t1d0s3
#-----/usr/local/bin/md5sum run on santesagne slice 4 --#
# md5sum /dev/dsk/c0t1d0s4
#987637b54899101a666239ed23282ff4 /dev/dsk/c0t1d0s4
#-----/usr/local/bin/md5sum run on santesagne slice 5 --#
# md5sum /dev/dsk/c0t1d0s5
#4f4b7ab62f99d95cdb67087d56936ac7 /dev/dsk/c0t1d0s5
#-----/usr/local/bin/md5sum not run on santesagne slice 6 slice zero size--#
#-----/usr/local/bin/md5sum run on santesagne slice 7 --#
# md5sum /dev/dsk/c0t1d0s7
#fc5e9e53bcd522a8c4b8e365846bf937 /dev/dsk/c0t1d0s7
#####
#-----md5sum on forensic system-----#####
#-----/usr/bin/md5sum run on forensic image slice0.dd --#
#[525]:~/SANS/practical/NRE_hack$> md5sum images/slice0.dd
1777bce42b21b70ea9885313d2f5f431 slice0.dd
#-----/usr/bin/md5sum run on forensic image slice1.dd --#
#[541]:~/SANS/practical/NRE_hack$> md5sum images/slice1.dd
2f50ef10a4bd36f7c3cca63f2be3fb31 slice1.dd
#-----/usr/bin/md5sum run on forensic image slice3.dd --#
#[546]:~/SANS/practical/NRE_hack$> md5sum images/slice3.dd
32cec9b9a21620d421d19a0bed4fef64 slice3.dd
#-----/usr/bin/md5sum run on forensic image slice4.dd --#
#[571]:~/SANS/practical/NRE_hack$> md5sum images/slice4.dd
987637b54899101a666239ed23282ff4 slice4.dd
#-----/usr/bin/md5sum run on forensic image slice5.dd --#
#[589]:~/SANS/practical/NRE_hack$> md5sum images/slice5.dd
4f4b7ab62f99d95cdb67087d56936ac7 slice5.dd
#-----/usr/bin/md5sum run on forensic image slice7.dd --#
#[592]:~/SANS/practical/NRE_hack$> md5sum images/slice7.dd
fc5e9e53bcd522a8c4b8e365846bf937 images/slice7.dd
#####
```

Media Analysis

In order to assist in evaluating the data several tools are present on the forensic system in addition to the standard utilities available on most Linux systems (e.g. grep, strings). Prominent in these tools are two developed by Brian Carrier of the @stake organisation. Task 1.60 is a series of forensic tools that can be used for analysing Microsoft and UNIX systems. Task uses code from the The Coroners Toolkit (TCT) with support added for FAT and NTFS. The second tool is recommended for use with TASK and was also developed by Brian Carrier. Autopsy 1.70 is the latest version (at the time of writing) of a forensic browser providing a graphical interface to the tools of TASK and also providing some automation of tasks.

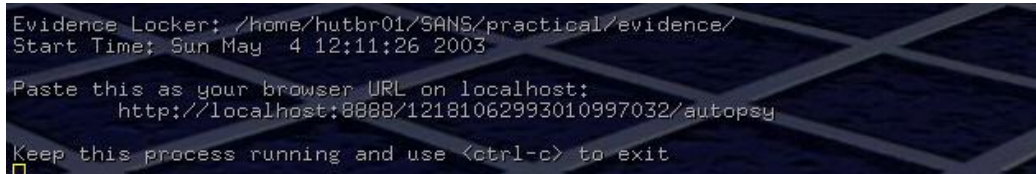
To be able to access the images from the device /dev/hdc1, it was mounted at /mnt/hack using the following command

```
$> sudo mount -ro,nodev,noexec,noatime /dev/hdc1 /mnt/hack
```

The Autopsy daemon is now started to listen on port 8888

```
$> sudo ./autopsy 8888 localhost
```

A url is printed to the screen by the autopsy process and this url is pasted into a web browser.



```
Evidence Locker: /home/hutbr01/SANS/practical/evidence/
Start Time: Sun May 4 12:11:26 2003

Paste this as your browser URL on localhost:
http://localhost:8888/12181062993010997032/autopsy

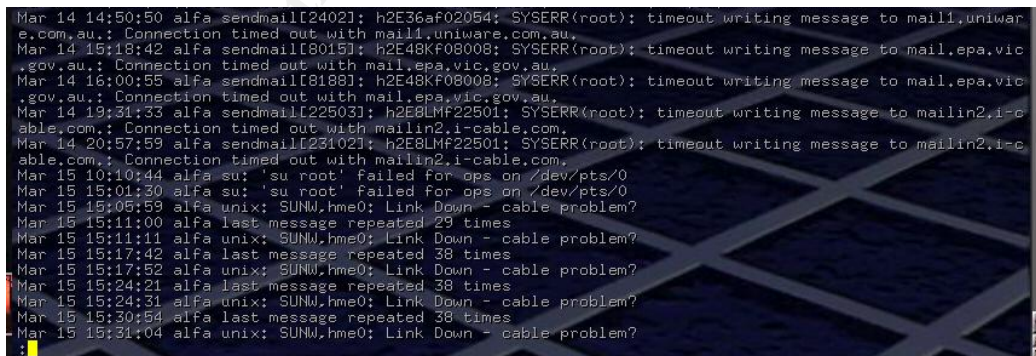
Keep this process running and use <ctrl-c> to exit
```

Through the browser a new case is opened this creates a directory ACME in the evidence directory where autopsy holds its cases. The host DNS_1 is added to the case. The timezone is set to the system time. The images should now manually be added to the evidence/ACME/DNS_1/images directory however autopsy wants to write to this directory with some of its findings making it impractical to have the directory read-only. A simpler alternative is to create symbolic links from the images at /mnt/hack to the evidence/ACME/DNS_1/images directory. This method was chosen and the images for slice0, slice 3, slice4, and slice 5 (/ , usr, var and opt) were added through the Autopsy browser. The SWAP partition slice 1 could not be added to this version of Autopsy.

To check for overt signs of access the following files were examined.

The relevant messages file. In solaris messages are stored in /var/adm. A check of the directory shows a new messages file was started on Mar 9 and ended on mar 16th this file was messages.0.gz. Autopsy allows for the export of files to another directory for viewing, this is necessary for compressed files especially. Using zcat piped to less one can read the file

```
#> zcat images-slice4.dd-var.adm.messages.0.gz|less
```



```
Mar 14 14:50:50 alfa sendmail[2402]: h2E36af02054: SYSERR(root): timeout writing message to mail1.uniwar
e.com.au.: Connection timed out with mail1.uniware.com.au.
Mar 14 15:18:42 alfa sendmail[8015]: h2E48KF08008: SYSERR(root): timeout writing message to mail.epa.vic
.gov.au.: Connection timed out with mail.epa.vic.gov.au.
Mar 14 16:00:55 alfa sendmail[8188]: h2E48KF08008: SYSERR(root): timeout writing message to mail.epa.vic
.gov.au.: Connection timed out with mail.epa.vic.gov.au.
Mar 14 19:31:33 alfa sendmail[22503]: h2E8LMP22501: SYSERR(root): timeout writing message to mailin2.i-c
able.com.: Connection timed out with mailin2.i-cable.com.
Mar 14 20:57:59 alfa sendmail[223102]: h2E8LMP22501: SYSERR(root): timeout writing message to mailin2.i-c
able.com.: Connection timed out with mailin2.i-cable.com.
Mar 15 10:10:44 alfa su: 'su root' failed for ops on /dev/pts/0
Mar 15 15:01:30 alfa su: 'su root' failed for ops on /dev/pts/0
Mar 15 15:05:59 alfa unix: SUNW.hme0: Link Down - cable problem?
Mar 15 15:11:00 alfa last message repeated 29 times
Mar 15 15:11:11 alfa unix: SUNW.hme0: Link Down - cable problem?
Mar 15 15:17:42 alfa last message repeated 38 times
Mar 15 15:17:52 alfa unix: SUNW.hme0: Link Down - cable problem?
Mar 15 15:24:21 alfa last message repeated 38 times
Mar 15 15:24:31 alfa unix: SUNW.hme0: Link Down - cable problem?
Mar 15 15:30:54 alfa last message repeated 38 times
Mar 15 15:31:04 alfa unix: SUNW.hme0: Link Down - cable problem?
```

As can be seen the only messages during the period in question refer to a mail misconfiguration, a failed attempt at su to root by ops after the event and complaints about the cable being disconnected.

The next file examined was the /etc/passwd this reveals user accounts on the system and the shells used. Root uses bash so the /.bash_history was examined. It revealed no evidence of someone trying to takeover the system.

The wtmp file shows a list of users logged in since it was created running the last command on the forensic system with the `-f` flag provides a sensible output.

```
#> last -f images-slice4.dd-var.adm.wtmp.raw
```

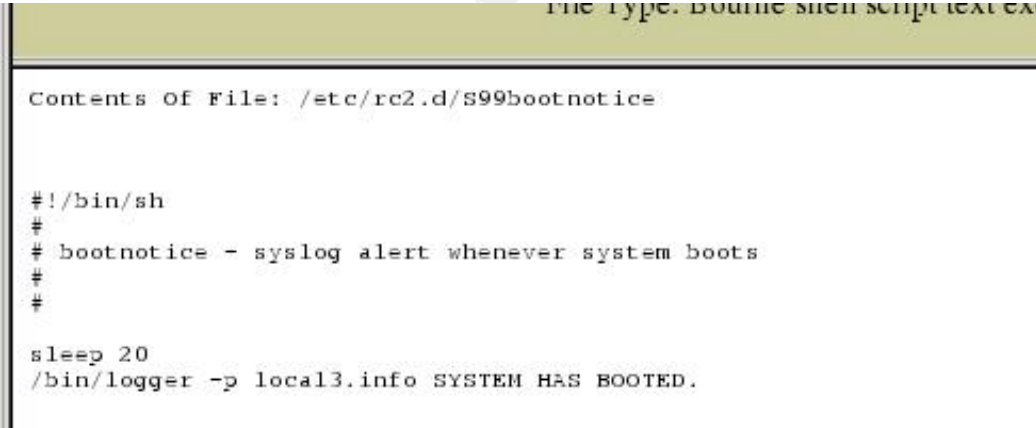
This file showed no one had logged onto the system during the period of time for which the alert was raised. The utmp and utmpx files were likewise checked for similar results. While it is possible for a hacker to zero these files or through a loadable kernel module hide their actions completely the lack of any sign of access is significant.

Given that the Apache web servers presence was raised as an indicator of a compromise the `/var/log/apache/access` file was checked. The access log revealed consistent activity for the past six months and no activity on the night of the incident.

The named configuration files in `/opt/local/etc` were checked for unusual entries.

An attacker will often alter or add system startup files to the host in an effort to change the function of the system. Solaris stores these files in `/etc/init.d` with links to the `/etc/rc.d` directories.

An examination of the directory `/etc/init.d` did not reveal any unusual files. Examination of the files within it showed them all to be either standard Solaris startup files in one case a `isc bind` startup file an Apache startup file and a `sshd` startup file. A check of `/etc/rc0.d`, `/etc/rc1.d`, `/etc/rc2.d` and `/etc/rc3.d` revealed only one other file which was a file created in 1997 for logging that the system was booting to `/var/adm/messages`.



```
Contents of File: /etc/rc2.d/S99bootnotice

#!/bin/sh
#
# bootnotice - syslog alert whenever system boots
#
sleep 20
/bin/logger -p local3.info SYSTEM HAS BOOTED.
```

(Graphic of Autopsy : author Brian Carrier)

The `/dev/` `/dev/devices` and `lib` directories were checked for Trojan files and unusual file or directory names like `"..."` or executable binaries or scripts and archive files.

SetUID and SetGID files allow processes to run with root privilege. In Solaris by default there are a large number of these files. By executing the command `#> awk 'f none [246] {print}'`

`/var/sadm/install/contents/var/sadm/install/contents` for SETUID SETGID files a listing of the operating systems default can be obtained. This was then compared with the files found in the `/sbin`, `/usr/sbin`, `/usr/bin`, `/usr/local/bin` and `/usr/local/sbin` directories. Additionally as other directories were checked permissions were noted in case of a rogue executable. None were found.

Solaris keeps its kernel parameters in /etc/system this file was also checked for untoward entries.

Finally cron and at were checked first for added entries to cron.allow and at.allow and also for the root crontab (there were no at entries)

```

Contents of File: /var/spool/cron/crontabs/root

#
#
# Housekeeping
# -----
0 * * * * /opt/local/sbin/rout
5 * * * * /opt/local/sbin/rout -c24 -s24 /var/tmp
10 * * * * /usr/lib/sendmail -q > /dev/null 2>&1
#
# Daily and weekly jobs
# -----
0 1 * * 0 /opt/local/sbin/agelogs -c /var/adm/messages 8
5 1 * * 0 /opt/local/sbin/agelogs -c /var/adm/sulog 8
# 10 1 * * 0 /opt/local/sbin/agelogs -c /var/cron/log 8
15 1 * * 0 /opt/local/sbin/agelogs -c /var/log/admin.log 8
20 1 * * 0 /opt/local/sbin/agelogs -c /var/log/authlog 8
25 1 * * 0 /opt/local/sbin/agelogs -c /var/log/bash.log 8
30 1 * * 0 /opt/local/sbin/agelogs -c /var/log/ftp.log 8
35 1 * * 0 /opt/local/sbin/agelogs -c /var/log/ipp.log 8
40 1 * * 0 /opt/local/sbin/agelogs -c /var/log/sshd.log 8
45 1 * * 0 /opt/local/sbin/agelogs -c /var/log/syslog 8
#
# We need to rotate root's mail, this is where postmaster mail goes - yeah, this is bad.
#
55 1 * * 0 /opt/local/sbin/agelogs -c /var/mail/root 8
#
# Any patches or packages to install
# -----
12 5 * * 1-5 if [ -f /etc/rc2.d/s77patch -o -f /etc/rc2.d/s77packages ]; then reboot; fi

```

(Graphic of Autopsy : author Brian Carrier)

The crontabs for adm and majordomo were also checked for discrepancy.

Timeline Analysis

To create the timeline of activity on the disk through Autopsy (or TCT for that matter) a body file must be created for each specific filesystem represented by the respective images. (e.g. slice0, slice3). This file contains extracted meta-data from the image. There are 3 type of files that can be extracted

Allocated Files files that have an allocated structure and are still in use

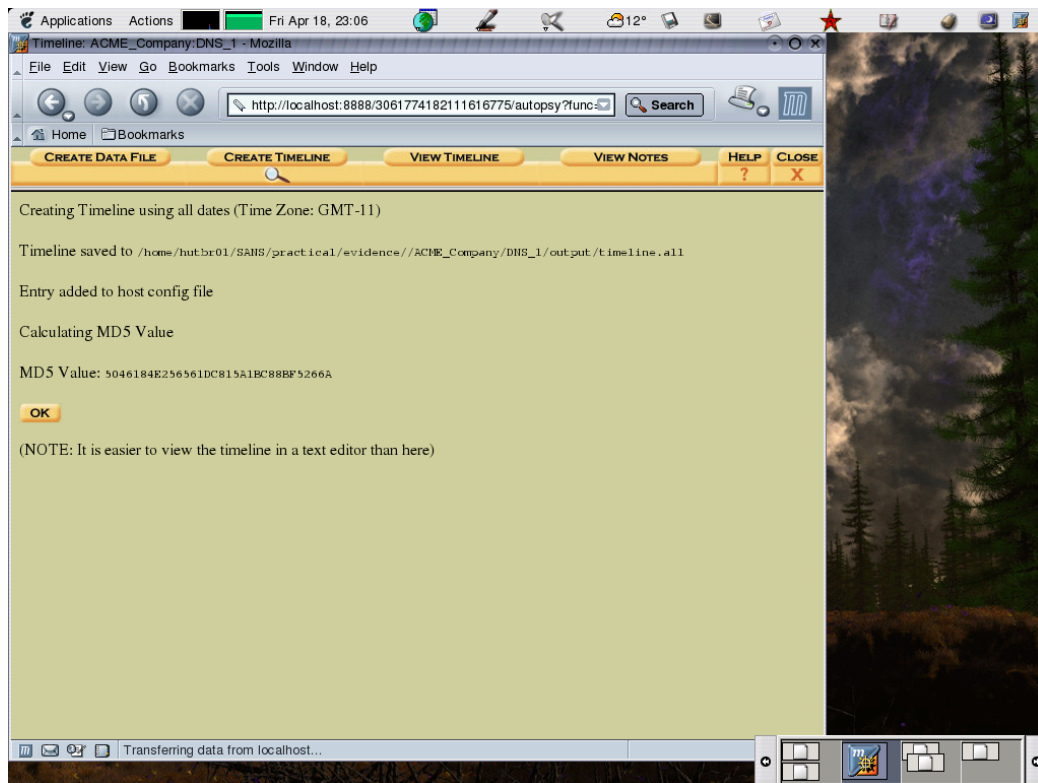
Unallocated Files deleted files that have not been overwritten yet

Unallocated Inodes files that have been deleted.

The Solaris implementation of the UFS filesystem removes the link between the file name and meta data structure and the link between meta data structure (inode) and the fragments. Unallocated data can only be looked at fragment by fragment which can take a long time. It would seem prudent therefore to examine other areas first for signs of intrusion.

Using the body file as its input a timeline can be created between two dates.

Normally a timeline would be from the month before the event to the present however for the purposes of the practical a far larger timeline is used, this will take in the systems life until the present.



(Graphic of Autopsy : author Brian Carrier)

The timeline created shows the first entry as 0831 Jan 18 1996. This is presumably when the system was first installed probably with Solaris 2.6. Further packages were installed in October of that year. In June 20 1997 another installation was carried out on the system followed by further work on July 16 of the same year. On February 09 1999 Perl was installed onto the system for the first time. The next day (Feb 10) GNU utilities (probably from sunfreeware.com) were installed onto the system in /opt/local/bin. ISC's bind binaries (like named) were also installed at this time. This activity might suggest that the system was changing roles and or a new administrator had taken over the system. This maybe when the system was first used as a name server. The next point of interest is on October 20 1999 when patches were added for Checkpoint Firewall-1. Obviously in the preceding months Checkpoint has been installed however the files have since been overwritten hence the absence from the timeline. February 08 2000 the system had Solaris operating systems installed. November 08 2000 appears to have had an upgrade to the operating system attempted (SUN pkgs and patches accessed) and when this fails an install from scratch occurs. The /var/sadm/system/logs/begin.log_2000_11_08 is started and the /lost+found, /usr/lost+found and /var are created. (Solaris 8 is probably installed here) Perl is again installed. November 15 2000 finds apache being installed. This is important to note given the incident report raised apache's presence as an indicator of an intrusion.

The system may have been given some form of mail list functionality on December 24th as a large number of now deleted files in /opt were created by user majordomo a common username for mail list software. Considering the

date this may relate to another system failing and DNS_1 being used in a temporary capacity.

An Oracle username exists in the password file and on Jan 30 2001 the /var/opt/oracle/oratab file was modified and changed

Packages are removed on November 8 2001 this is determined by the depend directories of several packages being accessed before their removal.

On Sunday March 15 2002 /var/log/XX and /var/log/XX2 were created this is worth noting due to the non standard naming used that may cause a false identification during an investigation.

January 21 2003 saw pkginfo command run on the system to give an indication of the packages installed on the system (SUN only).

On March 13 2003 at 1725 named was started for the last time before the incident.

On March 14 now deleted files were accesses from 1200 to 1300 and then again from 1500 till 2300.

On the day of the incident (March 15) at 00:08 /var was again accessed and certain now deleted files were also modified and changed. This pattern continued at 00:17,01:44,02:35,02:52, 05:27,0600, 0615. This last time is significant as this is when the alert was first raised by the IDS system of a TSIG attack. The activity continued at 0624, 0650, 0702,0745 and 0750. It is noticeable at this point that while still not aware of the files involved there does seem to be a periodic nature to the activity which may suggest system activity rather than an intruder. According to the Incident Response document at approximately 1000 the Administrators arrived on site and at 1009 we see the /var/log/apache/access directory checked. Then is appears a backup was attempted as large quantities of files in the /var/log area were accessed. This action affects the timeline and makes determining activities prior to the backup more difficult to determine.

At 1502 files were modified created and accessed in the /opt /partition.

Unfortunately these files have been deleted. The large groups of files deleted on the system make the forensic process far more difficult and raise questions as to why this took place on the system. Having only had limited access to the administrator concerned one can only surmise an element of panic and inexperience was involved. At 1504 /var/tmp was cleared out. Again this may have contained useful files.

Following some more file access activity by the administrators in the /var area (again deleted) the administrators left the system at around 1730 this is confirmed by the Incident Response document. At 1716 the utmp and wtmp files were modified and changed indicating a logout.

The system now isolated continued its normal system activity for the next few days until on Wednesday 18 at 1413 the system was brought to a halt and the disks removed for copying and analysis.

Recover Deleted Files

The Solaris implementation of the UFS Filesystem removes the link between deleted filenames and metadata so that normal methods of recovering deleted files by name is impractical. Investigation of the timeline is required to get a list of unallocated inodes deleted around the incident time. This list can now be recovered using autopsy. The period of time chosen was between 0001

on the 15/03 to 1730 on the 16th after the administrators had logged out of the system this is to take into account the large numbers of files deleted by the administrator team.

Using an editor to view the timeline we can see that the only activity at around the time of the incident was in slice4 (/var). A list of deleted files is apparent at this time

```

Sat Mar 15 2003 05:28:02 0 ..c ----- root/smtp other 142666 <slice4.dd-dead-142666>
Sat Mar 15 2003 06:00:58 0 ..a ----- root/smtp other 142653 <slice4.dd-dead-142653>
0 ma ----- root/smtp other 142647 <slice4.dd-dead-142647>
0 ..a ----- root/smtp other 142649 <slice4.dd-dead-142649>
0 ..a ----- root/smtp other 142650 <slice4.dd-dead-142650>
Sat Mar 15 2003 06:00:59 0 m.. ----- root/smtp other 142649 <slice4.dd-dead-142649>
Sat Mar 15 2003 06:01:02 0 m.. ----- root/smtp other 142650 <slice4.dd-dead-142650>
Sat Mar 15 2003 06:01:17 0 ..c ----- root/smtp other 142647 <slice4.dd-dead-142647>
0 ..c ----- root/smtp other 142649 <slice4.dd-dead-142649>
0 m.. ----- root/smtp other 142650 <slice4.dd-dead-142650>
0 ..c ----- root/smtp other 142653 <slice4.dd-dead-142653>
0 m.. ----- root/smtp other 142650 <slice4.dd-dead-142650>
Sat Mar 15 2003 06:15:44 0 ma ----- root/smtp other 142720 <slice4.dd-dead-142720>
Sat Mar 15 2003 06:15:52 0 ..c ----- root/smtp other 142720 <slice4.dd-dead-142720>
Sat Mar 15 2003 06:19:06 0 ..a ----- root/smtp other 142721 <slice4.dd-dead-142721>
Sat Mar 15 2003 06:19:07 0 ma ----- root/smtp other 142693 <slice4.dd-dead-142693>
Sat Mar 15 2003 06:19:22 0 m.. ----- root/smtp other 142721 <slice4.dd-dead-142721>
0 ..c ----- root/smtp other 142721 <slice4.dd-dead-142721>

```

One notes that in the 4th column the filesize is zero this is due to the aforementioned procedure of Solaris in unlinking deleted filenames and metadata. The data from these files cannot be directly recovered through autopsy. The only practical method left is to string search the data. Navigating to the keyword search menu there is an option to create an unallocated data file (dls) from the slice image (in this case slice4)(dd) This is needed in order to be able to analyse the deleted data.

The screenshot shows the Autopsy software interface with the 'KEYWORD SEARCH' menu open. The 'Generate Strings File' and 'Create Unallocated Data File' options are visible. The 'File Name' field for the unallocated data file is set to 'slice4.dls'. The 'Generate MDS?' checkbox is checked for both options. The 'EXTRACT UNALLOCATED' button is highlighted.

(Graphic of Autopsy : author Brian Carrier)

The slice4.dd.str can now be searched for suspicious keywords or clues to any intrusion. Autopsy will give a data unit number beside found occurrences of the string in question

As an example from slice5.dd.str a search for the string “hacker” returns the data unit 155976, using autopsy one can check on the original fragment 627904. As explained before only show fragments rather than complete files.

```

      ASCII (display - report) - Hex (display - report) - Strings (display - report)
      File Type: a /usr/local/bin/perl script text executable

Fragment 627904
Not Allocated
Group: 21
Inode not found

-----
ASCII Contents of Fragment 627904 (1024 bytes) in images/slice5.dd

#!/usr/local/bin/perl
#
# Non-parsed headers CGI 1.1 error script in Perl to handle error requests
# from NCSA HTTPd 1.4 via ErrorDocument. This should handle all errors in
# almost the same fashion as NCSA HTTPd 1.4 would internally.
#
# This script is in the Public Domain. NCSA and the author offer no
# guarantee's nor claim any responsibility for it. That's as pseudo-legalise
# as I get.
#
# This script doesn't do any encryption or authentication, nor does it
# contain hooks to do so.
#
# This was written for Perl 4.016. I've heard rumours about it working with
# other versions, but I'm no Perl hacker, so how would I know?
#
# Brandon Long / NCSA HTTPd Development Team / Software Development Group
# National Center for Supercomputing Applications / University of Illinois
#
# For more information:
# NCSA HTTPd : http://hoohoo.ncsa.uiuc.edu/docs/
# CGI 1.1 : http://hoohoo.ncsa.uiuc.edu/cgi/
# ErrorDocument : http://hoohoo.ncsa.uiuc.edu/docs/setup/srm/ErrorDocument.html
# Example CGI : http://

```

(Graphic of Autopsy : author Brian Carrier)

This is clearly part of an error handling script to assist with an http process. To get the whole file would require looking at the rest of the data fragments in the slice..

String Search

While one can search for strings using the autopsy browser it is quicker to do so from the command line. A file (hackterms) was created to include several likely keywords whose presence can be checked for through the grep command.

```

[564]:~/SANS/practical/evidence/ALME_Company/DNS_1$ cat hackterms
rootkit
hack
bot
sniff
backdoor
trojan
Own
lrk
knark
adore
slapper
promisc
gcc
ssh
kill

```

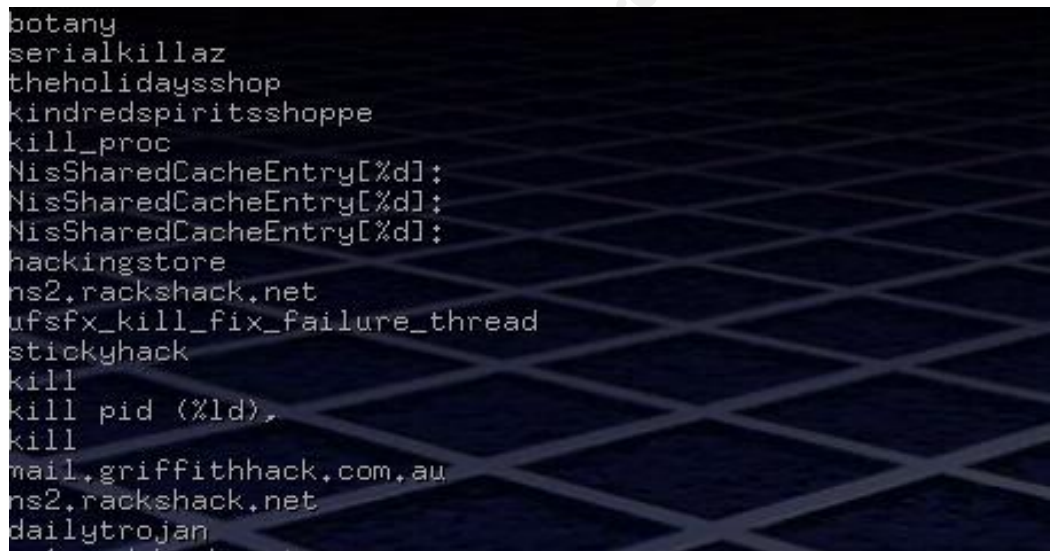
The strings chosen for the file were selected for the likelihood of being left behind in a compromise “rootkit”, “lrk”, “knark” “adore” and “slapper” are references to rootkits and were obtained from <http://www.chkrootkit.org/> a website offering a tool for checking for these kits. “hack” “backdoor” “Own” and “trojan” are common terms used in system cracking. “kill” is a command for controlling processes a cracker may use it to stop, pause or start system programs. Sshd is used to send encrypted traffic, gcc is the GNU compiler used to compile many programs. Finally “sniff” and “promisc” refer to the system potentially having a network sniffer installed.

The SWAP slice (slice1) was checked for these strings using the command `#strings -a output/slice1.dd|grep -i -f hackterms > swap_strings`

The output of the file swap_strings was further restricted following an initial viewing by exempting strings containing the usernames botta and killan, the search engine hotbot and the term robot.

`#less swap_strings|grep -v -i botta|grep -v -i killan|grep -v -i hotbot|grep -v -i robot|more`

This output was found to contain a series of mail files, a large number of DNS requests and some entries referring to the kill command.



```
botany
serialkillaz
theholidayssshop
kindredspiritsshoppe
kill_proc
NisSharedCacheEntry[%d]:
NisSharedCacheEntry[%d]:
NisSharedCacheEntry[%d]:
hackingstore
ns2.rackshack.net
ufsfx_kill_fix_failure_thread
stickyhack
kill
kill pid (%ld).
kill
mail.griffithhack.com.au
ns2.rackshack.net
dailytrojan
```

Suspicious words were checked through the ubiquitous search engine google.com. These queries were negative. For instance hackershardware refers to Xbox cheats while dailytrojan.com refers to an American baseball team. (this query may well have been made as a joke.)

Serialkillaz is a site providing software license cracks. Rackhack is a rack supply company. Stickyhack is a website containing “cheats” for games like diablo

Amongst the other data found was EMAC buffer commands, ssh key generation and some directory listings.

It was concluded there was no evidence of an intrusion in the strings output of the swap partition.

As stated earlier the activity at the time of and immediately prior to the IDS alert took place in the /var partition now imaged as slice4.dd. This partition

Then the following command is run

The output was however unreadable.

Looking at the slice4.dls file through a pager (less) does reveal some readable data along with large sections of seemingly random data. Obviously none of the keywords were in the viewable fragments and are only found in the random strings that are part of the other data.

The reason for the random data is difficult to explain. Some tests on the data reveal it was not compressed and is in fact appears to be encrypted. This combined with the lack of knowledge about the data fragments mean that this data cannot realistically be examined.

A reasonable conclusion to this investigation is that the system was not in fact compromised. The timeline shows activity in the /var/ partition on the night of the attack with the files affected being deleted after the event by the administrator.

While this and the fact that the machine was shutdown before live system processes could be captured made this examination difficult. However the absence of untoward activity in the SWAP strings output. The benign nature of the messages and lastlog files that should have indicated unusual access and the fact that none of the accounts appear to have been accessed during the period of the attack all point to this being a false alarm resulting from an alert by the IDS and some inexperience from the System Administrator involved. These views were expressed to the ACME security officer along with some recommendations on how to deal with a system that may have been compromised. The recommendations concerning preservation of system state have since been adopted.

Appendix1 InfoLeak Attack

Bugtraq ID2321 Description

“BIND is a server program that implements the domain name service protocol. It is in extremely wide use on the Internet, in use by most of the DNS servers.

It is believed that most (if not all) versions of BIND in use contain a vulnerability that may allow an attacker to view named's memory.

This may aid an attacker in further attacks.”

CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND

Original release date: January 29, 2001

Last revised: --

Source: CERT/CC

A complete revision history can be found at the end of this file.

Systems Affected

Domain Name System (DNS) Servers running various versions of ISC BIND (including both 4.9.x prior to 4.9.8 and 8.2.x prior to 8.2.3; 9.x is not affected) and derivatives. Because the normal operation of most services on the Internet depends on the proper operation of DNS servers, other services could be impacted if these vulnerabilities are exploited.

Three of these vulnerabilities (VU#196945, VU#572183, and VU#868916) were discovered by the COVERT Labs at PGP Security, who have posted an advisory regarding these issues at <http://www.pgp.com/research/covert/advisories/047.asp>

The fourth vulnerability (VU#325431) was discovered by Claudio Musmarra.

The Internet Software Consortium has posted information about all four vulnerabilities at <http://www.isc.org/products/BIND/bind-security.html>

Description

U#325431 - Queries to ISC BIND servers may disclose environment variables

This vulnerability is an information leak in the query processing code of both BIND 4 and BIND 8 that allows a remote attacker to access the program stack, possibly exposing program and/or environment variables. This vulnerability is triggered by sending a specially formatted query to vulnerable BIND servers.

Impact

VU#325431 - Queries to ISC BIND servers may disclose environment variables

This vulnerability may allow attackers to read information from the program stack, possibly exposing environment variables. In addition, the information obtained by exploiting this vulnerability may aid in the development of exploits for VU#572183 and VU#868916.

Appendix 2 TSIG Attack

CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND

Original release date: January 29, 2001

Last revised: --

Source: CERT/CC

A complete revision history can be found at the end of this file.

Systems Affected

Domain Name System (DNS) Servers running various versions of ISC BIND (including both 4.9.x prior to 4.9.8 and 8.2.x prior to 8.2.3; 9.x is not affected) and derivatives. Because the normal operation of most services on the Internet depends on the proper operation of DNS servers, other services could be impacted if these vulnerabilities are exploited.
infrastructure.

Three of these vulnerabilities (VU#196945, VU#572183, and VU#868916) were discovered by the COVERT Labs at PGP Security, who have posted an advisory regarding these issues at

<http://www.pgp.com/research/covert/advisories/047.asp>

The fourth vulnerability (VU#325431) was discovered by Claudio Musmarra.

The Internet Software Consortium has posted information about all four vulnerabilities at <http://www.isc.org/products/BIND/bind-security.html>

VU#196945 - ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code

During the processing of a transaction signature (TSIG), BIND 8 checks for the presence of TSIGs that fail to include a valid key. If such a TSIG is found, BIND skips normal processing of the request and jumps directly to code designed to send an error response. Because the error-handling code initializes variables differently than in normal processing, it invalidates the assumptions that later function calls make about the size of the request buffer.

Once these assumptions are invalidated, the code that adds a new (valid) signature to the responses may overflow the request buffer and overwrite adjacent memory on the stack or the heap. When combined with other buffer overflow exploitation techniques, an attacker can gain unauthorized privileged access to the system, allowing the execution of arbitrary code.

Bibliography and References

Attack Advisories

Cert Coordination Center <http://www.cert.org/>

Bugtraq <http://www.securityfocus.com/archive/1>

Rootkit Detection

<http://www.chkrootkit.org/>

Task and Autopsy

Author Brian Carrier (carrier@sleuthkit.org)

www.atstake.com/research/tools/forensic/

<http://sleuthkit.sourceforge.net/autopsy/index.php/>

Sun Solaris

Solaris Manual Pages various

<http://www.sun.com/solutions/blueprints/>

Solaris File deletion

<http://www.honeynet.org/scans/scan15/proj/bc/>

E. Eugene Schultz, Russell Shumway "Incident Response A Strategic Guide to Handling System and Network Security Breaches" New Riders 2001

Kevin Mandia and Chris Prosise "Incident Response: Investigating Computer Crime" Osborne/McGraw-Hill 2001

Part 3: Legal Issues of Incident Handling

In the prescribed scenario I take the role of a system administrator for an Internet Service Provider that provides access to paying customers.

I receive a telephone call from a law enforcement officer who informs me that an account on your system was used to 'hack' into a government computer. He asks me to verify the activity by reviewing the logs under my control and determine if those logs reflect whether or not the activity was initiated there or from an upstream provider.

I review the logs and can only determine a valid user account logged in via a dialup account during the period of the suspicious activity.

Note: For the purpose of the scenario the identity of the law enforcement officer is presumed to have been verified and that this exercise does not involve social engineering.

A What if any, information can you provide to the law enforcement officer over the phone during the initial contact?

I reside in the state of Victoria in Australia and as such must comply with both state and federal legislation.

In addition working for an ISP I am in all likelihood bound by the Australian Communications Industry Forum's relevant Industry Code (ACIF C5327:2001).

The Information Privacy Principle 11 of the Federal Privacy Act 1988 does not permit the disclosure of personal information held by an organization. While a log is unlikely to contain much personal information a case could possibly be made that a person's activities on the Internet are of a personal nature.

Under the Telecommunication Act 1997 (Commonwealth) it is an offence under part 13 of the Act for an employee of an ISP to disclose the contents of communications carried by the ISP's systems and network infrastructure, (in this case information from the logs would reveal the content of the communication).

However on request I must provide what is termed in Part 14 of the Act 'reasonably necessary' help to law enforcement and national security agencies. This request can be made in one of two ways, firstly the agency can certify that assistance is required. This is done through production of a certificate and does not require any judgment on the part of the ISP. This is documented under section 282(3) of the Telecommunications Act 1997. The second option from the law enforcement officers, covered under section 282(1) of the same Act is to provide a written request that describes the

offence being investigated. In this situation section 3.12 of the Australian Communications Authority 'Telecommunications and Law Enforcement Manual' 1988 provides a set of guidelines for making an assessment of whether the request meets the criteria of 'reasonably necessary'. In this scenario to describe the offence a state law enforcement officer may be able to refer to the Victorian Crimes (Property Damage and Computer Offences) Act 2003 sections:

- (247B) Unauthorised access, modification or impairment with intent to commit serious offence.
- (247D) Unauthorised impairment of electronic communication
- (247G) Unauthorised access to or modification of restricted data.

A federal officer can refer to the Cybercrime Act 2001 in particular sections:

- (477.1) Unauthorised access, modification or impairment with intent to commit serious offence.
- (477.3) Unauthorised impairment of electronic communication
- (478.1) Unauthorised access, modification or impairment of electronic communication where the person is not entitled to cause that access.
- (478.2) Unauthorised access to, or modification of, restricted data

The exception to the notion of 'reasonably necessary' assistance would be if the law enforcement officer was from the Australian Security Intelligence Organisation (ASIO) in which case a written authorisation from the ASIO Director General is all that is required. This is described in section 283 of the Telecommunications Act 1997.

All these sections do require some form of documentation to be produced therefore until I, or my management receive the appropriate documentation I am not able to provide information to the officer.

B What must the law enforcement officer do to ensure you to preserve this evidence if there is a delay in obtaining any required legal authority

There is no provision for the preservation of data without a legal authority. (One would assume that the data would be saved as part of the ISP's archiving procedure)

Agencies may request that I keep certain information pertaining to a user but this must be through the mechanism of 'reasonably necessary' help as defined in the previous answer.

As I am not allowed to access what may be evidence under the aforementioned Privacy Act (principle 11) and the Telecommunications Act 1997 Part 13 I cannot at this point in time ensure the preservation of the data.

C What legal authority if any, does the law enforcement officer need to provide you in order for you to send him your logs?

As defined in Part 13 of the Telecommunications Act 1997 and section 3LA of the Crimes Act 1914 (amended as part of the Cybercrime Act 2001), the law enforcement officer can request customer information, in this case logs from me through a search warrant obtained from a court with jurisdiction in this area, (given the delays inherent in obtaining a search warrant this step is unlikely to be taken).

The officer can provide a written request that sets out the offence being investigated and enough information so I (and or my senior management) can decide the request is 'reasonably necessary'. (Refer to Telecommunications Act 1997 Section 282(1)).

The Australian Communications Authority has produced a 'Determination of Requirements' for ISP's to use in making judgments on written requests. (Australian Communications Authority 'Telecommunications and Law Enforcement Manual' 1988 Section 3.12) it requires that requests meet the following criteria:

The request must be in writing and should be dated

The document should contain the law enforcement agency's logo or letterhead.

An officer of the law enforcement agency should personally sign the document

A priority or time frame for executing the request should be indicated

The services required to be performed by the ISP should be detailed

The offence being investigated is cited ("the information is reasonably necessary for the investigation of an offence contrary to section ____ of the Act of 19____.")

"A signed assurance that the information will be used only for the purpose for which it was sought and that it will be secured against unauthorised disclosure"

The Law enforcement agencies fax number

Or the officer can provide a certificate stating that obtaining the logs is 'reasonably necessary' assistance. (Refer to Telecommunications Act 1997 Section 282(3)).

The certificate can come from the Victorian state police, the Australian Federal police, the National Crime Authority or a "prescribed authority established under a law of the commonwealth or state" (Australian Communications Authority 'Telecommunications and Law Manual' 1988 Section 3.10). The certificate should:

"Identify the person making the certification and specify that the person is authorized

Identify the agency

Include the date of the issue of the certificate

Specify where the information is to be sent

Specify which provision (ss. 282(3), (4) or (5) of the Act) is being relied upon" (Australian Communications Authority 'Telecommunications and Law Manual' 1988 Section 3.7).

If the request was from the Australian Security Intelligence Organisation then a separate procedure must be followed.

ASIO does not use the concept of 'reasonably necessary assistance'. In order for me to provide him with the logs he would need to notify me that providing the logs is necessary for the ASIO to perform its functions (or would be in the future) and that he has been authorised in writing by the Director General of Security to receive the log files.

If any of these documents are produced by the law enforcement agency then a record of the handing over of the logs to the enforcement agency (unless the agency was ASIO) must be made by me in order to comply with Part 13 of the Telecommunications Act 1997.

D What other investigative activity are you permitted to carry out at this time?

There are no explicit restrictions referenced in the Telecommunications Act 1979 or the Telecommunications Interception Act 1979 however if contained within the request for 'reasonably necessary' assistance extra tasks such as checking other logs for further evidence may be undertaken.

Section 2.9.8 of the Australian Communications Industry Forum's Industry Code (ACIF C5327:2001) notes that providing information about how long records are kept and whether various logs are current or archived can be considered part of 'reasonable assistance'.

However if the request was limited to just obtaining the log file then this would be the only other direct assistance I could give.

Best practice would suggest that letting the agency drive the process would help ensure the investigation is carried out in a correct manner from a legal point of view and that the evidence is preserved intact. Searching logs for addresses however innocuous in motive could result in a legal defence that the logs have been tampered with and damage or impede the case made by the law enforcement agency.

E How would your actions change if your logs disclosed a hacker gained unauthorized access to your system at some point, created an account for him/her to use, and used THAT account to hack into the government system?

Part 14 of the Telecommunications Act 1997 requires that the ISP and employees including myself do their best to prevent the ISP's infrastructure from being used in the commission of offences against the state of Victoria or the Australian Commonwealth. In the previous scenario the user under investigation had a legitimate account. In this scenario there is a situation of unauthorized access to a system under the ISP control which is then used to hack into a government system.

If the disclosure was as a result of the law enforcement officer's request for 'reasonably necessary' assistance then the actions taken would be

unchanged except for the ISP Computer Systems Incident Response Team (CSIRT) would be notified to handle the breach in security.

However if the log entries were disclosed by myself or fellow employees through the execution of our normal duties then the appropriate law enforcement agency would be notified in compliance with section 3.4 of the Australian Communications Authority 'Telecommunications and Law Enforcement Manual' 1988. The act of contacting the agency would be through a predefined point of contact as recommended in Section 3.15 of the Australian Communications Authority 'Telecommunications and Law Enforcement Manual' 1988 and in Section 2.9.1 of the Australian Communications Industry Forums Industry Code (ACIF C5327:2001). The ISP CSIRT would be notified and the evidence would be preserved as an image with the appropriate chain of custody document.

References

Privacy Act 1988 incorporating amendments up to Act No. 125 of 2002

Telecommunications Act 1997

Act No. 47 of 1997 as amended, incorporating amendments up to Act No.140 of 2002

Telecommunications (Interception) Act 1979

Act No. 114 of 1979 as amended, incorporating amendments up to Act No. 125 of 2002

Telecommunications Interception Legislation Amendment Act 2002

No. 67, 2002

Australian Communications Industry Forum

Industry Code ACIF C5327:2001 –Provision of Assistance to National Security, Enforcement and Government Agencies

<http://www.efa.org.au/Issues/Privacy/surveillance.html#tia>

http://www.aca.gov.au/aca_home/licensing/telcomm/app_form/law.htm

CyberCrime Act 2001

<http://scaleplus.law.gov.au/html/pasteact/3/3486/pdf/161of2001.pdf>

Australian Communication Authority "Telecommunications and Law Enforcement Manual" 1998

http://www.aca.gov.au/aca_home/licensing/radcomm/about_radcomms_licensing/leac.pdf

Crimes (Property Damage and Computer Offences) Act 2003

Crimes (Property Damage and Computer Offences) – Introduction Print

<http://www.dms.dpc.vic.gov.au/pdocs/bills/B01382/B01382I.html>

Model Criminal Code Discussion Paper January 2000

© SANS Institute 2003, Author retains full rights.