Fight crime.
Unravel incidents... one byte at a time.

GCFA Practical

Version 1.1b

Abstract: Analysis of an unknown binary provided by GIAC, followed by a forensic analysis on a system and a Canadian perspective surroundiung the legal issues of incident handling. (Part 1, Part 2 Option 1 & Part 3)

By Christopher Lee
April 16th, 2003

Table of Content

Legend

Throughout this paper, a number of typographical conventions have been used.

Ordinary text is in Times New Roman 12 point font, as is this sentence.

The Times New Roman 10 point font is used to denote text typed into and printed on a UNIX shell terminal, source code, and other text files on a UNIX system. Text boxes are often used to set these types of text apart:

```
[root@machine00]# echo "text the user types is shown in blue"
commands the user types are shown in blue
[root@machine00]# echo "command output remains in black font"
command output remains in black font
```

Boxes like this are used to denote comments from the author

Part 1 – Analyze and Unknown Binary

Configuration of the Forensic Lab:

The forensic lab is constructed via the use of the VMWare Workstation version 3.1, where a virtual networking environment was created with no contact to the "real world". VMWare Workstation is software written by VMWare Inc. designed to create "virtual" computers within a host system, and it's designed and generally accepted as an alternative from having to configure multiple physical computer systems.

The VMWare virtual adapter on host machine was enabled with no network protocols nor services bind to it. A network Sniffer (Ethereal) was running on the host machine listening to the traffic of the VMWare virtual adapter.

The Linux workstation, where the bulk of the analysis was taking place, was built directly from the RedHat 7.3 distribution CDs (where each CD's integrity has been validated with MD5 checksums from the vendor) as a minimal installation with no additional patches. The following tools necessary for the forensic analysis was added to the installation from trusted sources after validations of their respective MD5 checksums.

1. strace, from strace-4.4.4.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. This is a tool to provide a trace of a given software process on all APIs called, software library accessed, file system access and etc.
2. strings, from binutils-2.11.93.0.2-11.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. This tool parses through a given file and extracts all ASCII text strings embedded in the file.
3. unzip, from unzip-5.50-2.i386.rpm on the disc 1 of the RedHat 7.3 distribution CDs. This tool decompresses (or unzips) files previously been compressed (or zipped) using PKZIP/WinZip compatible algorithm.
4. zip, from zip-2.3-12.i386.rpm on the disc 1 of the RedHat 7.3 distribution CDs. Similar to the "unzip" tool, this is the tool that does the actual "zipping" of the files.
5. make, from make-3.78.1-8.i386.rpm on the disc 1 of the RedHat 7.3 distribution CDs. "make" is software compilation interface that was designed to make ease of the configuration, the compilation and the installation of software from its source code files.
6. glibc libraries, from glibc-devel-2.2.5-34.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. This set of libraries contains the most commonly utilized C functions often used by developer to simply their development efforts.
7. kernel headers for glibc, from glibc-kernheaders-2.4-7.14.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. The "kernel headers" are header files necessary for the compilation of certain C source codes.
8. cpp, from cpp-2.96-110.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. "cpp" is the C Pre-compilation Processor required by the gcc compiler below.

9. gcc, from gcc-2.96-110.i386.rpm on the disc 2 of the RedHat 7.3 distribution CDs. "gcc" is the GNU version of the C compiler and it's used to compile C programs from their source codes into binaries executable in the present environment.

Binary Details:

Upon retrieving the unknown binary file (binary_v1.1.zip) from GIAC website on Dec 25 of 2002, I first attempted to answer the following 6 questions about this unknown binary:

1. File name

   This was easy enough; "atd" is the name of this binary. That seem to suggest this is either the scheduler daemon commonly found on *nix systems, or a program was designed to make others think that.

2. MAC Time

   Using the "debugfs –R "stat <682>" /dev/sda2" command (where 682 is the inode number of the "atd" file), I was able to obtain the mtime, atime and the ctime fairly easily.

Mtime (last modification time):	Thu, Aug 22, 14:57:54, 2002
Atime (last access time):	Thu, Aug 22, 14:57:54, 2002
Ctime (last change time):	Fri, Dec 27, 15:36:31, 2002

   Note: Since the binary was extracted from the zip file on Dec 27, the ctime was "updated" to reflect that.

3. File Owners

   Though I tried to unzip the zip file with the –X option, it appeared that this zip file was created from DOS/Windows operating system where the file ownership info was not included as part of the file attributes. This could have been easily avoided if the Trojan was first preserved using a common Unix/Linux program "tar" prior to the transport of the binary to a remote system, which would have retained the ownership info as well as the MAC times.

4. File Size

   As revealed by the "ls -l" command, the file "atd" has a size of 15,348 bytes.

5. MD5 Hash

The command "md5sum atd" yields the MD5 checksum hash of "48e8e8ed3052cbf637e638fa82bdc566 atd", which is identical to the content of

```
$ md5sum ./atd
48e8e8ed3052cbf637e638fa82bdc566 atd
$ cat atd.md5
48e8e8ed3052cbf637e638fa82bdc566 atd
```

the "atd.md5" file.

6. Key Words found within the program

The command "strings atd" revealed the following ASCII text strings within the atd file:

/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
```

Program Description

Based on the information gathered previously (as documented in the above section), I know this "atd" program was last accessed/executed on Thu, Aug 22nd of 2002, and that it generates certain type(s) of network traffic (based on output of the "strings" command) and the string "LOKI2 route [(c) 1997 guild corporation worldwide]" is probably a suggestion on the identity of this mystery binary.

The command "file atd" yields the following output:

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

This further confirms the executable nature of this program, and tells us that it is a 32-bit ELF binary compiled for Intel 80386 CPU.

The command "ldd atd" was not able to yield us with useful information on the libraries this binary was linked with (with obscured error message about "no such file or directory"). However, by peeping into the first few hundred bytes of the file (using the "hexdump" command), /lib/ld-linux.so.1 and /lib/libc.so.5 were seemingly the dynamic libraries it was linked with. "ld-linux.so.1" was the dynamic library loader commonly found on pre-RedHat 7.0 (i.e. RedHat 6.2 and earlier) distribution of the Linux operating system, and libc.so.5 were the version 5 of the libc library, which was the standard library for RedHat 2.0 through 4.2 releases of Linux distribution.

Base the output of the "strings" command obtained earlier, this program seems to create network sockets of certain type (e.g. the keywords "socket" and "setsockopt"), and possibly also create/modify certain files (e.g. the keyword "umask"). Interestingly, this program also possesses the ability to duplicate another instance of itself during execution (e.g. the keyword "fork").

Summarizing what I have ascertained thus far, the best description of this file is as follows:

This file "atd" is a Linux executable, written in the C programming language and was compiled with linkages to "ld-linux.so.1" and "libc.so.5" (dynamic library loader and the library itself for libc version 5); this is probably a Trojan designed for older Linux systems, with ability to create or manipulate network sockets and files. This unknown

binary also possesses the ability to create a running duplicate of itself, under certain circumstances.

Finally, it is possibly the (or a variant of) version 2 of the infamous LOKI Trojan. This file was last executed at 2:57pm on Thursday, August 22nd of 2002.

LOKI2 is an information-tunneling program. It is a proof of concept work intending to draw attention to the insecurity that is present in many network protocols. In this implementation, we tunnel simple shell commands inside of ICMP_ECHO / ICMP_ECHOREPLY and DNS namelookup query / reply traffic. To the network protocol analyzer, this traffic seems like ordinary benign packets of the corresponding protocol. To the correct listener (the LOKI2 daemon) however, the packets are recognized for what they really are.[1] However, in practice, LOKI2 has been seen to provide intruders with covert channels into a compromised system (since most system administrator would simply "ignore" ICMP/DNS packets into their networks).

Forensic Details

Binary Analysis on a RedHat 7.3 Linux system:

To add the support for both libc5 and its dynamic loader, I added both the libc-5.3.12-31.i386.rpm and ld.so-1.9.5-13.i386.rpm to the forensic workstation. This ought to be enough to get this binary running in this lab environment.

I first tried to run the "strace" command (i.e. "strace –tf –ff –otrace-atd ./atd") as an ordinary user and got the following error message before the program terminated:

```
[clee@dhcppc0 clee]$ strace -tf -ff -otrace-atd ./atd

[fatal] invalid user identification value: Success
```

Looking into the trace-atd file, I noticed the program was trying to obtain both the "real" and the "effective" user ID of the process "atd". Perhaps this binary requires the privilege of a particular user??

I then tried to execute this as the root user. This time, the program ran and spits out the following message on the screen:

```
[root@dhcppc0 /root]# strace -tf -ff -otrace-atd ./atd

LOKI2   route [(c) 1997 guild corporation worldwide]
Process 5364 attached
```

[1] Quoted directly from Volume 7, Issue 51 of the Phrack Magazine

With the program still running, I examined the trace files from another session. Two trace files were created: one for the "atd" process itself (i.e. trace-atd) and the other is for a process it forked (i.e. trace-atd.5364).

- "trace-atd"
  I looked for all incidences of "open" or "write" throughput the trace file, and concluded this program does not write to any files on the local file system, though it attempted to open from a few of them (i.e. /usr/i486-linux-libc5/lib/libc.so.5, /etc/ld.so.cache and /usr/share/locale/en_US.iso885915/LC_MESSAGES) with only the READ flag and it tried also to ascertain the existence of the following locale files:
  - /etc/locale/C/libc.cat
  - /usr/lib/locale/C/libc.cat
  - /usr/lib/locale/libc/C
  - /usr/share/locale/C/libc.cat
  - /usr/local/share/locale/C/libc.cat

  It also opened two IPv4 sockets. One of them is an ICMP socket, while the other is a raw socket on port 255.

  Shortly after it forked, this process terminated on its own.

- "trace-atd.5364"
  The program appeared to try, unsuccessfully, to open /dev/tty for both read and write. It then changed its current working directory to /tmp and changed its umask to 022. Finally it listens on the ICMP socket its parent process created, and terminates on its own after 1 hour (3600 seconds).

Base on the information provided in the trace files (trace.atd and trace-atd.5364), no file system footprints were observed. The only changes noticed was the two raw sockets opened on the system during the time when "atd" and forked copy was running.

Obviously, this is very odd, given the suspected Trojan nature of this unknown binary. There is a good likelihood that certain external interaction is necessary for this unknown binary to demonstrate its capabilities. The two network sockets it created are probably where this program expects the external interactions.

Further, as discovered later on during the attempts to locate and compile the potential source code of this unknown binary, this Trojan was compiled with C libraries found on RedHat 4.2 or similar Linux systems (ones that supports Linux kernel version 2.0.x). Perhaps a better or more accurate way to assess the behaviour of this unknown binary is to do so from an environment for which it was compiled in and therefore designed to operate with.

Binary Analysis on a RedHat 4.2 Linux system:

Another Linux VMWare virtual machine is then built based on RedHat 4.2 distribution stored on ftp.dulug.duke.edu, one of RedHat Linux' official mirror sites. The content of the files used for this installation were validated through its MD5 checksums. The unknown binary "atd" is then moved to this new environment. The MD5 checksum of this binary is then computed, after transfer to the new environment, and verified against the original MD5 checksum to ensure the file was not tampered with during transit.

I first tried to run this unknown binary as an ordinary user:

```
[clee@dhcppc0 clee]$ strace -tf -ff -otrace-atd-42 ./atd

[fatal] invalid user identification value: Unknown error
```

Then I run it as the root user:

```
[root@dhcppc0 /root]# strace -tf -ff -otrace-atd-42 ./atd

LOKI2   route [(c) 1997 guild corporation worldwide]
Process 32159 attached
```

Thus far, I have confirmed this binary's requirement to be executed with root user privilege.

Now, let's look at the trace files created by the above command.

- "trace-atd-42"

  By examine the content of the trace-atd-42 (from the last command) and the previous trace-atd file (obtained from running the same binary on the RedHat 7.3 forensic workstation), I have concluded both file contains the essentially the same trace outputs, with obvious differences in the time, process ID and semaphore information.

- "trace-atd-42.32159"

  In comparison with the file "trace-atd.5364", something interesting/different was observed here. Instead of simply terminating itself after one hour (3600 seconds), it actually restarted itself for another 3600 seconds.

However, we are still not getting any interesting insight into the operation or the purpose of this Trojan. We will proceed with external interactions previously discussed.

Binary Analysis using Normal ICMP Traffic as the External Interaction:

The first attempt is to see how this unknown binary reacts when ordinary ICMP packets (such as ICMP-echo, as commonly generated by the "PING" command) were sent to those sockets.

As in previous attempts, the command "strace –tf –ff –otrace-atd-42 ./atd" is used to launch the unknown binary and to capture its interaction with the host system. A sequence of ICMP-echo packets was delivered from a remote system to this host after the unknown binary has been executed. The "atd" process was then terminated with a "kill" command.

Similar to our prior observation, the main process terminates shortly after it forked a copy of itself. By examine the content of the child process' trace file ("trace-atd-42.26086"), the regular ICMP-echo packet does not seem to trigger any response from the "atd" process.

Binary Analysis using LOKI2 Traffic as the External Interaction:

Finally, the LOKI2 client, obtained through the compilation process from the next section, is used to interact with this suspected LOKI2 server.

Once again, the command "strace –tf –ff –otrace-atd-42 ./atd" is used to launch the unknown binary and to capture its interaction with the host system. From another session on the same computer, the command "./loki –d 127.0.0.1" was executed and the following output was captured from the command prompt:

```
[root@dhcppc0 /root]# ./loki -d 127.0.0.1

LOKI2   route [(c) 1997 guild corporation worldwide]
loki> ls
install.log
libtermcap-2.0.8-4.i386.rpm
passwd-0.50-7.i386.rpm
rdate-0.960923-1.i386.rpm
loki>
```

At this point, the LOKI2 client was terminated with a CTL-C break. The strace command was then terminated with a "kill" command.

Four (4) trace files were created by strace as the result of this attempt: "trace-atd-42", "trace-atd-42.28418", "trace-atd-42.28422", and "trace-atd-42.28423". From analyzing the content of these four files, the following behaviour is observed.

1. The 1st child process listens for incoming ICMP messages.
2. Once the "right" kind of message is received (such as the "ls" command from the LOKI2 client, in this case), a 2nd child process is then forked.

3. The 2nd child process then forked a 3rd child process to execute the "ls" command as the root user (which is the owner of the "atd" process) via the "sh" shell, which, in this case, is the Bourne-Again Shell (i.e. bash). The output of the "ls" command is then passed back to the 2nd child process to send back to the LOKI2 client.
4. Both the 3rd and the 2nd child processes then terminated themselves once the command is executed.

Footprints?

Based on our binary analysis, this Trojan does not modify any files on the file system and thus limited forensic info is available to identify the intruder. However, the following footprints should be visible on systems infected with this Trojan:

1. The file "atd" with the file size 15,348 byte and MD5 checksum of "48e8e8ed3052cbf637e638fa82bdc566".
2. The command "netstat –na " ought to reveal two additional raw socket listeners on port 1 and port 255.
3. Assuming the intruder planted this "atd" Trojan to start automatically with system (e.g. insert it into one of the system init scripts), then:
    a. The message "LOKI2 route [(c) 1997 guild corporation worldwide]" might be visible in the /var/log/message file.
    b. The keyword "atd" might be visible in one of the init scripts (at /etc/rc.d/init.d/ or one of the /etc/rc.d/rc#.d directories) or in one of the scripts called by those init scripts.
4. The command "ps –aux | grep atd | grep –v grep" should return information about a process name "atd".
5. Since commands issued from the LOKI2 client is executed as the root user via its shell, those commands might still be available within the command history of the root user.

Obviously, remote/local access to this host with the super-user (root) access is required to plant this Trojan. Reasonably, other forensic trails ought to be available to show how this intruder gained the necessary access to the infected system.

Program Identification

Thus far, this unknown binary seems to LOKI2. A copy of the source code was extracted (using the extract.pl tool on the same page) from issue 51 of Phrack Magazine (available via http://www.phrack-dont-give-a-shit-about-dmca.org/show.php?p=51&a=6). Initial attempts to compile the source code on the RedHat 7.3 machine were unsuccessful.

I noticed the source code was published on September 1997; perhaps the C library has changed drastically since??

The source code is then moved to a machine running RedHat 4.2 (release on April 1997) and attempted another compilation with the default "Makefile" configuration. This time, the source code was compiled with no errors.

The file size of the captured binary is 15,348 bytes; this is exactly the same as the compiled LOKI2 server binary.

The command "md5sum atd" reveals the MD5 checksum of the captured binary:
48e8e8ed3052cbf637e638fa82bdc566

The command "md5sum lokid" reveals the MD5 checksum of the compiled LOKI2 server binary:
48e8e8ed3052cbf637e638fa82bdc566

By the virtue of having the exact same file size and MD5 checksum, I was able to ascertain the identity of the captured previously unknown binary to that of a LOKI2 server compiled with the same "Makefile" configuration as published on the issue 51 of the Phrack Magazine.

Legal Implications

Obviously, the trivial ways to ascertain the use of this program on this system is to look for one of the footprints documented in the "Forensic Detail" section of this write-up.

One other way of ascertaining if this binary was executed on the system is through the examination of its MAC times. If the program was executed, then its access time (atime) ought to be updated to when it was last executed, which should be after its modified time (mtime). This analysis, however, will only work if someone had not manipulated the MAC times of this file.

Given the nature of this Trojan, which is to provide covert channel for someone to access this system remotely. It is very likely for the user of this Trojan not being the legitimate privileged user of this system, and more importantly, someone would have to first obtain the "root" privilege to "install" this Trojan at the first place. If it has been determined that someone has inappropriate obtained the "root" privilege on this machine, it will constitute the violation of Section 342.1 of Criminal Code of Canada, which states

> Every one who, fraudulently and without colour of right,
> **(a) <u>obtains, directly or indirectly, any computer service,</u>**
> (b) by means of an electro-magnetic, acoustic, mechanical or other device, intercepts or causes to be intercepted, directly or indirectly, any function of a computer system,
> (c) uses or causes to be used, directly or indirectly, a computer system with intent to commit an offence under paragraph (a) or (b) or an offence under section 430 in relation to data or a computer system, or

> (d) uses, possesses, traffics in or permits another person to have access to a computer password that would enable a person to commit an offence under paragraph (a), (b) or (c)
>
> is guilty of an indictable offence and liable to imprisonment for a term not exceeding ten years, or is guilty of an offence punishable on summary conviction.
>
> (Quoted from http://www.rcmp-grc.gc.ca/crim_int/hackers_e-a.htm#appendixA)

Interview Questions

Obviously, by now, the interviewee knew something is up (i.e. why am I being asked to be here?) and is probably going to be defensive on direct questions surrounding the incident. The strategy here would be to try wining him/her over by establishing some sort of connection and use that as a wedge to lower his/her guard and starting talking. The goal is obviously to get him/her to confess the crime or to (accidentally) provide more clues to the investigators.

Q1. I heard that you are really good with the computer stuff and knows all sorts of really cool tricks. I am really impressed. Where did you learn all that?

(assuming we received a half-decent response here, otherwise keep on chatting/identifying with him/her)

Q2. Wow, that's real cool. So, what do you with all those new cool tricks you know? I mean, if I were you, I will be dying to check out what all those tricks could do for me…

Q3. Now, hypothetically speaking, how would someone go about making sure he/she could always get back into a system without being noticed?

Q4. So, what were you doing at 2:57pm on August 22nd 2002?

Q5. How would you explain this network traffic report that documents a constant stream of "abnormal" IP packets between this system here and your computer?

Additional Information

Phrack Magazine Volume 7, Issue 51 September 01, 1997, article 06 of 17
http://www.phrack-dont-give-a-shit-about-dmca.org/show.php?p=51&a=6

Appendix A — Criminal Code of Canada offences related to hacking
http://www.rcmp-grc.gc.ca/crim_int/hackers_e-a.htm#appendixA

Criminal Code of Canada, [R.S. 1985, c. C-46]
http://www.canlii.org/ca/sta/c-46/

Part 2 – Option 1: Perform Forensic Analysis on a system

Synopsis of Case Facts

Breach of security on a database server in a local high school (XYZ high school) was discovered on October 15th, 2002. One of the teachers discovered the grades of several students were drastically higher than what was recorded on a prior physical printout. This database server situates behind a firewall only granting access from web servers on the local network. The local network is connected to the Internet directly through a router provided by the school board (no translation of addresses is implemented).

Unfortunately, no intrusion detection systems (IDS) were implemented at this location; therefore the investigator must proceed without any knowledge on the when or how this crime was committed. The firewall log showed no unusual connections to the database, thus the suspicion is one of the web servers has been compromised and used as a launch pad to crack the security on the database server.

As part of the post-intrusion response, the local system administrator shut down all the local web servers and submit their hard drives to the school board's forensic team for analysis. One of such servers is an old web server at the northwest corner of the high school's computer room.

Describe the system(s) you will be analyzing.

This is a Linux web server, providing part of the high school's web presence on the Internet, running version 1.3.22 of the Apache web server software. This server is a clone PC purchased from a local vendor and its hardware configuration is as follows:

- Asset Tag # XYZ-012
- AMD 486 DX2/80 CPU
- 24 MB of RAM
- 1 IDE 546MB Hard Drive
- 1 1.44 MB 3.5" Floppy Drive
- 1 quad-speed IDE CDROM drive
- 1 3Com 3C509 EtherLink II 10 Mbps RJ45 network adapter

The local system administrator, as outlined by the school board's incident response policy, shut down this server and turned its hard drive over to the investigation team after a written request was served.

Hardware

The hard drive from the confiscated machine is described as follows:

Tag# 01 Maxtor 7546AT Hard Drive, Serial # C6013JTS, Size 546 MB

Drive Geometry:			Jumper Configuration:		
Cylinder	Head	Sectors	Jumper	Master/Single	Slave
1060	16	63	J20	ON	OFF

Image Media

The hard drive was installed (but not mounted) as the second hard drive on the primary channel of a RedHat Linux workstation. I want to create reliable images of relevant partitions of this hard drive for analysis on the forensic workstation. The "fdisk –l /dev/hdb" command was used to display the partition table on this hard drive and the data on the partition was read by the trusty "dd" command (with the "if=" parameter to read from the appropriate partitions) and then transmitted to the forensic via a TCP connection created by the "nc" command[2]. MD5 (Message Digest version 5) checksum is then calculated via the command "md5sum" (often distributed as part of any Unix/Linux OS distributions) for each of these partitions for integrity validation by the receiving host.

```
[root@dhcppc0 /root]# fdisk –l /dev/hdb

Disk /dev/hdb: 32 heads, 63 sectors, 530 cylinders
Units = cylinders of 2016 * 512 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1         480      483808+  83  Linux
/dev/hdb2             481         530       50400   82  Linux swap
[root@dhcppc0 /root]# dd if=/dev/hdb1 | nc 192.168.254.10 8888
1069482+0 records in
1069482+0 records out
 punt!

[root@dhcppc0 /root]# md5sum /dev/hdb1
f79bf19e048d5feb3d4e3fdbddd85aa4          /dev/hdb1

[root@dhcppc0 /root]# dd if=/dev/hdb2 | nc 192.168.254.10 8888
1069482+0 records in
1069482+0 records out
 punt!

[root@dhcppc0 /root]# md5sum /dev/hdb2
5c4cac34514c2904b0732fc0b55139bc          /dev/hdb2
```

[2] NetCat, an arbitrary TCP/UDP listener and forwarder written by Hobbit.

On the receiving machine (i.e. our forensic workstation), the "nc" commands are utilized to redirect the data stream to files on the local file system. MD5 checksums are then calculated for those image files and compared with the MD5 checksum of the original partitions.

```
[root@forensic /root]# nc –l –p 8888 > /images/hdb1image.img

[root@forensic /root]# md5sum /images/hdb1image.img
f79bf19e048d5feb3d4e3fdbddd85aa4          /images/hdb1image.img

[root@forensic /root]# nc –l –p 8888 > /images/hdb2image.img

[root@forensic /root]# md5sum /images/hdb2image.img
5c4cac34514c2904b0732fc0b55139bc          /images/hdb2image.img
```

Since the image files possesses the same MD5 checksum as the hard drive partitions, we have ascertained the files "hdb1image.img" and "hdb2image.img" are good forensic images of the hard drive.

Media Analysis of System

Here comes the media analysis of the partitions on this hard drive. The goal here to look for the following "clues" in the file system that might provide insight to the investigator on how (if any) the crime was committed:

1. Modification to operating system software and configuration
2. Back doors, check for setuid and setgid files
3. Signs of a Sniffer program
4. History files
5. /proc examination
6. Start up files

To make sure we do not accidentally modify the partitions during the analysis, the MD5 checksum values of both partitions are first saves to the files /morgue/md5.txt and the options "ro, loop, nodev, noexec, nosuid, noatime" were supplied to the "mount" command. Those options are explained as follows:

- ro:
  Mount the file system read-only.
- loop:
  Mount on a loop device. The loop device is a device driver that allows an image file to be mounted as though it were a normal block device
- nodev:
  Do not interpret character or block special devices on the file system.
- noexec:
  Do not allow execution of any binaries on the mounted file system. This option might be useful for a server that has file systems containing binaries for architectures other than its own.
- nosuid:

Do not allow set-user-identifier or set-group-identifier bits to take effect. (This seems safe, but is in fact rather unsafe if you have suidperl(1) installed.)

- noatime:
  Do not update inode access times on this file system.

```
[root@forensic morgue]# md5sum hdb1image.img hdb2image.img > md5.txt
[root@forensic morgue]# mkdir /mnt/forensic
[root@forensic morgue]# mount –o ro,loop,nodev,noexec,nosuid,noatime /morgue/hdb1image.img
/mnt/forensic
[root@forensic morgue]# cd /mnt/forensic
[root@forensic forensic]# touch test-if-read-only
touch: creating `test-if-read-only': Read-only file system
[root@forensic forensic]# mount
/dev/sda2 on / type ext3 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/sda1 on /boot type ext3 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/cdrom on /mnt/cdrom type iso9660 (ro,nosuid,nodev)
/morgue/hdb1image.img on /mnt/forensic type ext3 (ro,noexec,nosuid,nodev,noatime,loop=/dev/loop0)
```

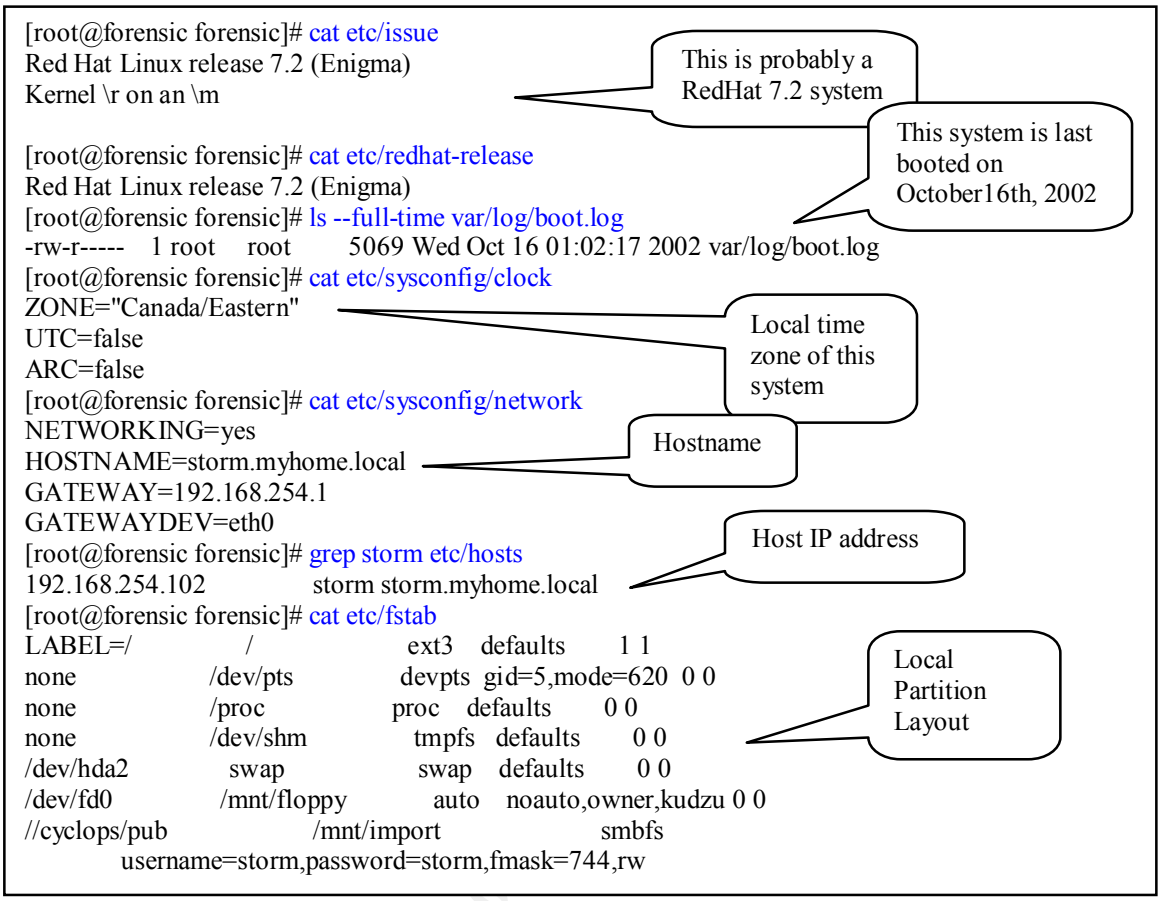
The file system is indeed read only

Confirmed the options supplied to the "mount" command

I then proceed to collect the following information about this system:

- OS and version: RedHat Linux 7.2
- Host's time zone: Canada/Eastern
- Hostname storm
- Host IP address(es) 192.168.254.100
- Last boot date: October 16th, 2003
- Partition map:

LABEL=/	/	ext	Defaults	1	1
None	/dev/pts	devpts	gid=5,mode=620	0	0
None	/proc	proc	Defaults	0	0
None	/dev/shm	tmpfs	Defaults	0	0
/dev/hda2	swap	swap	Defaults	1	1
/dev/fd0	/mnt/floppy	auto	noauto,owner,kudzu	0	0
//Cyclops/pub	/mnt/import	smbfs	username=storm,password=storm,fmask=744,rw		



Modification to operating system software or configuration

As previously established (through the /etc/issue and /etc/redhat-release files), this appears to be a RedHat Linux 7.2 based system. The "rpm -Va" (RedHat Package Manager) command is then used to verify the integrity of all system software and/or configuration files installed through RPM packages on this system. Since the system being analyzed in mounted on /mnt/forensic, the option "—root /mnt/forensic" is supplied to instruct the "rpm" command to treat the directory "/mnt/forensic" as the system's root directory during its validation process.

```
[root@forensic forensic]# rpm --root /mnt/forensic/ -Va | tee /tmp/verified.txt
.M......  /proc
.M......  /usr/local
.M......  /usr/local/bin
.......T c /etc/syslog.conf
missing   /dev/log
.M....G.  /dev/tty1
.M....G.  /dev/tty2
.M....G.  /dev/tty3
.M....G.  /dev/tty4
.M....G.  /dev/tty5
.M....G.  /dev/tty6
S.5....T c /etc/krb.conf
.....UG. c /var/lib/rpm/Filemd5s
.....UG. c /var/lib/rpm/Sha1header
.....UG. c /var/lib/rpm/Sigmd5
.......T c /etc/ssh/moduli
.......T c /etc/pam.d/sshd
.......T c /etc/rc.d/init.d/sshd
.......T c /etc/nscd.conf
.......T c /etc/rc.d/init.d/nscd
.......T c /etc/rpc
.......T c /etc/httpd/conf/ssl.crl/Makefile.crl
.......T c /etc/httpd/conf/ssl.crt/Makefile.crt
.......T c /etc/httpd/conf/ssl.crt/ca-bundle.crt
.......T c /etc/httpd/conf/ssl.crt/snakeoil-ca-dsa.crt
.......T c /etc/httpd/conf/ssl.crt/snakeoil-ca-rsa.crt
.......T c /etc/httpd/conf/ssl.crt/snakeoil-dsa.crt
.......T c /etc/httpd/conf/ssl.crt/snakeoil-rsa.crt
.......T c /etc/httpd/conf/ssl.key/snakeoil-ca-dsa.key
.......T c /etc/httpd/conf/ssl.key/snakeoil-ca-rsa.key
.......T c /etc/httpd/conf/ssl.key/snakeoil-dsa.key
.......T c /etc/httpd/conf/ssl.key/snakeoil-rsa.key
.......T c /etc/httpd/conf/ssl.prm/snakeoil-ca-dsa.prm
.......T c /etc/httpd/conf/ssl.prm/snakeoil-dsa.prm
missing   /var/cache/ssl_gcache_data.dir
missing   /var/cache/ssl_gcache_data.pag
missing   /var/cache/ssl_gcache_data.sem
.......T c /etc/krb5.conf
.......T c /etc/rc.d/init.d/kdcrotate
..5....T c /etc/mime.types
S.5....T  /etc/cron.daily/logrotate
S.5....T c /etc/xinetd.d/telnet
S.5....T c /etc/pam.d/system-auth
.......T c /etc/cron.daily/00webalizer
S.5....T c /etc/webalizer.conf
missing   /var/www/html/usage/msfree.png
missing   /var/www/html/usage/webalizer.png
S.5....T c /etc/openldap/ldap.conf
S.5....T c /etc/ntp.conf
S.5....T c /etc/inittab
```

```
Unsatisfied dependencies for perl-5.6.1-26.72.3: perl-CPAN, perl-CGI, perl-DB_File, perl-
NDBM_File
Unsatisfied dependencies for fetchmail-5.9.0-11: smtpdaemon
S.5....T   /boot/kernel.h-2.4.9
.......T c /etc/httpd/conf/access.conf
S.5....T c /etc/httpd/conf/httpd.conf
.......T c /etc/httpd/conf/magic
.......T c /etc/httpd/conf/srm.conf
.......T c /etc/logrotate.d/apache
.......T c /etc/rc.d/init.d/httpd
......G.   /usr/sbin/suexec
.....U..   /var/cache/httpd
.......T c /var/www/html/index.html
.......T c /etc/pam.d/sudo
S.5....T c /etc/sudoers
.......T c /etc/fdprm
.......T c /etc/pam.d/chfn
.......T c /etc/pam.d/chsh
.......T c /etc/pam.d/login
```

The output of the "rpm –Va" command is divided into three columns; the first column describes where the differences are, the second column denotes if the file is a "configuration file" (by the character "c") and the last column displays the name of the file/directory in question.

The symbols used in the first column is explained as follows[3]:

- S File Size differs
- M Mode differs (includes permissions and file type)
- D Device major/minor number mis-match
- 5 MD5 sum differs
- L readLink(2) path mis-match
- U User ownership differs
- G Group ownership differs
- T mTime differs

Generally speaking, I am not concerned with files whose mtime had changed as long as its MD5 checksums are still good (i.e. assuming the potential intruder did not modify the RPM's database files, however this is something to verify later on). To identify those files that has changed since installation, I used the "grep" command with its "-v" option to filter away messages about:

1. Only mTime was changed
2. Missing files
3. Unsatisfied dependancies

[3] Quoted from the man page for the rpm command, as written on June 6th, 2001

```
[root@forensic forensic]# cd /tmp
[root@forensic tmp]# grep -v '\.\.\.\.\.\.\.\.T' verified.txt | grep -v 'missing' | grep -v 'Unsatisfied'
.M......   /proc
.M......   /usr/local
.M......   /usr/local/bin
.M....G.   /dev/tty1
.M....G.   /dev/tty2
.M....G.   /dev/tty3
.M....G.   /dev/tty4
.M....G.   /dev/tty5
.M....G.   /dev/tty6
S.5....T c /etc/krb.conf
.....UG. c /var/lib/rpm/Filemd5s
.....UG. c /var/lib/rpm/Sha1header
.....UG. c /var/lib/rpm/Sigmd5
..5....T c /etc/mime.types
S.5....T   /etc/cron.daily/logrotate
S.5....T c /etc/xinetd.d/telnet
S.5....T c /etc/pam.d/system-auth
S.5....T c /etc/webalizer.conf
S.5....T c /etc/openldap/ldap.conf
S.5....T c /etc/ntp.conf
S.5....T c /etc/inittab
S.5....T   /boot/kernel.h-2.4.9
S.5....T c /etc/httpd/conf/httpd.conf
......G.   /usr/sbin/suexec
.....U..   /var/cache/httpd
S.5....T c /etc/sudoers
```

With that thought in mind, the files of interests would be:

- /etc/krb.conf
- /etc/mime.types
- /etc/cron.daily/logrotate
- /etc/xinetd.d/telnet
- /etc/pam.d/system-auth
- /etc/webalizer.conf
- /etc/openldap/ldap.conf
- /etc/ntp.conf
- /etc/inittab
- /boot/kernel.h-2.4.9
- /etc/httpd/conf/httpd.conf
- /etc/sudoers

All of the above files were examined manually and nothing out of ordinary were observed; all changes detected seemed to be the result of site-specific configurations.

At the same times, I also noticed the following:

- RPMs providing capabilities "smtpdaemon", "perl-CPAN", "perl-CGI", "perl-

DB_File", and "perl-NDBM_File" are missing.

- The following files are missing:
  - "/dev/log",
  - "/var/www/html/usage/msfree.png",
  - "/var/www/html/usage/webalizer.png",
  - "/var/cache/ssl_gcache_data.dir",
  - "/var/cache/ssl_gcache_data.pag" and
  - "/var/cache/ssl_gcache_data.sem".
- Files/directories with changed group, user memberships and/or permissions:
  - /proc
  - /usr/local
  - /usr/local/bin
  - /dev/tty1
  - /dev/tty2
  - /dev/tty3
  - /dev/tty4
  - /dev/tty5
  - /dev/tty6
  - /var/lib/rpm/Filemd5s
  - /var/lib/rpm/Sha1header
  - /var/lib/rpm/Sigmd5
  - /usr/sbin/suexec
  - /var/cache/httpd

Once again, none of these observations lead to any noteworthy discoveries; they are different simply as the result of the system boot-up or intentional modifications by the system administrator.

Back doors, check for setuid and setgid files

Ensuring they could get back into the compromised systems, backdoors are typically implemented as soon as the intruder gained access to the system. The backdoors are typically covert channels that will give the intruder the "root" access (also known as the super-user access) upon connection (with or without further authentication), and these covert channels are almost always provided through executables with "setuid" and/or "setgid" bits enabled.

The "find" command commonly found on Linux/Unix systems is the perfect tool for locating files with special attributes. Its extensive parameters and the ability to create logical relationship between parameter makes it the ideal method of locating files with either "setuid" or "setgid" bits enabled. In this case, the "-perm" parameter is used to specify the "setuid" (i.e. permission value of 0x004000 in hexadecimal) and "setgid" (i.e. 0x002000 in hexadecimal) permissions we are looking for, the "-type f" parameter is supplied to narrow our searches to only files, and the "-ls" parameter instructs the "find" command to list the file attributes of the matched files.

```
[root@forensic forensic]# find . \( -perm -004000 -o -perm -002000 \) -type f -ls
 36335  795 -rws--x--x   2 root     root       808822 Feb 20  2002 ./usr/bin/suidperl
 36339   35 -rwsr-xr-x   1 root     root        34476 Aug 27  2001 ./usr/bin/chage
 36341   37 -rwsr-xr-x   1 root     root        36208 Aug 27  2001 ./usr/bin/gpasswd
 36645   38 -rwsr-xr-x   1 root     root        37528 Jan 17  2002 ./usr/bin/at
 36440   14 -rwxr-sr-x   1 root     mail        12500 Jun 30  2001 ./usr/bin/lockfile
 36489   26 -rwxr-sr-x   1 root     slocate     25020 Jun 25  2001 ./usr/bin/slocate
 36335  795 -rws--x--x   2 root     root       808822 Feb 20  2002 ./usr/bin/sperl5.6.1
 36524   15 -r-s--x--x   1 root     root        13476 Aug  7  2001 ./usr/bin/passwd
 36557    7 -r-xr-sr-x   1 root     tty          6444 Aug 28  2001 ./usr/bin/wall
 36646   14 -rws--x--x   1 root     root        13164 Jun 24  2002 ./usr/bin/chfn
 36564   14 -rws--x--x   1 root     root        12512 Jun 24  2002 ./usr/bin/chsh
 36582    6 -rws--x--x   1 root     root         5488 Jun 24  2002 ./usr/bin/newgrp
 36593    9 -rwxr-sr-x   1 root     tty          8768 Jun 24  2002 ./usr/bin/write
 36601   22 -rwsr-xr-x   1 root     root        21280 Jun 24  2001 ./usr/bin/crontab
 36519  235 -rwsr-xr-x   1 root     root       239024 May 22  2002 ./usr/bin/ssh
 36717   82 ---s--x--x   1 root     root        82348 Apr 22  2002 ./usr/bin/sudo
 42374   20 -rwsr-xr-x   1 root     root        18444 Aug 27  2001 ./usr/sbin/ping6
 42378   10 -rwsr-xr-x   1 root     root         9804 Aug 27  2001 ./usr/sbin/traceroute6
 42407    7 -rwxr-sr-x   1 root     utmp         6604 Jun 24  2001 ./usr/sbin/utempter
 70878  317 -rwxr-sr-x   1 root     503        321358 Jul 10  2002 ./usr/sbin/postdrop
 42402    7 -rwsr-xr-x   1 root     root         6340 Feb  6  2002 ./usr/sbin/usernetctl
 42468   21 -rwsr-xr-x   1 root     root        20120 Jun 25  2001 ./usr/sbin/traceroute
 42473   12 -r-s--x---   1 root     48          11308 Jun 19  2002 ./usr/sbin/suexec
 70879  281 -rwxr-sr-x   1 root     503        284356 Jul 10  2002 ./usr/sbin/postqueue
 84740   24 -rwsr-xr-x   1 root     root        23436 Aug 27  2001 ./bin/ping
 84859   58 -rwsr-xr-x   1 root     root        57628 Jul 24  2001 ./bin/mount
 84860   29 -rwsr-xr-x   1 root     root        28380 Jul 24  2001 ./bin/umount
 84890   20 -rwsr-xr-x   1 root     root        18452 Jul 23  2001 ./bin/su
 77044   16 -r-sr-xr-x   1 root     root        15088 Nov  9  2001 ./sbin/pwdb_chkpwd
 77045   18 -r-sr-xr-x   1 root     root        16824 Nov  9  2001 ./sbin/unix_chkpwd
 77144    5 -rwxr-sr-x   1 root     root         4120 Feb  6  2002 ./sbin/netreport
```

A quick search revealed the following files with “setuid” and/or “setgid” bits enabled:

- /usr/bin/suidperl
- /usr/bin/chage
- /usr/bin/gpasswd
- /usr/bin/at
- /usr/bin/lockfile
- /usr/bin/slocate
- /usr/bin/sperl5
- /usr/bin/passwd
- /usr/bin/wall
- /usr/bin/chfn
- /usr/bin/chsh
- /usr/bin/newgrp
- /usr/bin/write
- /usr/bin/crontab
- /usr/bin/ssh

- /usr/bin/sudo
- /usr/sbin/ping6
- /usr/sbin/traceroute6
- /usr/sbin/utempter
- /usr/sbin/postdrop
- /usr/sbin/usernetctl
- /usr/sbin/traceroute
- /usr/sbin/suexec
- /usr/sbin/postqueue
- /bin/ping
- /bin/mount
- /bin/umount
- /bin/su
- /sbin/pwdb_chkpwd
- /sbin/unix_chkpwd
- /sbin/netreport

Base on the output of running the same command on a freshly-built (from known-good source media) identically configured RedHat 7.2 system, I know the above list of SUID/SGID binaries are nothing out of ordinary.

I then check the "/etc/passwd" (i.e the list of accounts) and "/etc/shadow" (i.e. the actual password file) files for more conventional type of backdoors; the rogue accounts.

```
[root@forensic forensic]# ls -l etc/passwd
-rw-r--r--    1 root     root         1347 Aug  3  2002 etc/passwd
[root@forensic forensic]# ls -l etc/shadow
-r--------    1 root     root          918 Aug  3  2002 etc/shadow
```

Base on the above finding, the permissions for those files, along with the ownership and its dates seem to be in good order. Let's take a look at their contents.

```
[root@forensic forensic]# cat etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/var/spool/news:
```

```
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/dev/null
rpm:x:37:37::/var/lib/rpm:/bin/bash
rpc:x:32:32:Portmapper RPC user:/:/bin/false
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/bin/false
apache:x:48:48:Apache:/var/www:/bin/false
advu:x:500:500::/home/advu:/bin/bash
bbuser:x:501:501::/home/bbuser:/bin/bash
postfix:x:502:502::/no/where:/no/shell
ntp:x:38:38::/etc/ntp:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
sshd:x:74:74::/var/empty/sshd:/bin/false
dnscache:x:503:504:DJBDNS dnscache user:/no/where:/no/shell
dnslog:x:504:505:DJBDNS dnslog user:/no/where:/no/shell
```

Almost every accounts are assigned with dummy shells[4], with the exception of "root", "sync", "shutdown", "halt", "news", "rpm", "advu", and "bbuser" users. Users "sync", "shutdown", "halt" and "news" are system accounts and are configured to launch their functional binaries as their shells. That leaves us with users "root", "rpm", "advu" and "bbuser" as the only ones with interactive shells. We shall move onto the "/etc/shadow" file for additional clues.

```
[root@forensic forensic]# cat etc/shadow
root:$1$/X4bav3e$K2RhyREqxWEaXdbDXUcGW0:11672:0:99999:7:::
bin:*:11672:0:99999:7:::
daemon:*:11672:0:99999:7:::
adm:*:11672:0:99999:7:::
lp:*:11672:0:99999:7:::
sync:*:11672:0:99999:7:::
shutdown:*:11672:0:99999:7:::
halt:*:11672:0:99999:7:::
mail:*:11672:0:99999:7:::
news:*:11672:0:99999:7:::
uucp:*:11672:0:99999:7:::
operator:*:11672:0:99999:7:::
games:*:11672:0:99999:7:::
gopher:*:11672:0:99999:7:::
ftp:*:11672:0:99999:7:::
nobody:*:11672:0:99999:7:::
mailnull:!!:11672:0:99999:7:::
rpm:!!:11672:0:99999:7:::
rpc:!!:11672:0:99999:7:::
rpcuser:!!:11672:0:99999:7:::
```

[4] Shells like "/sbin/nologin", "/no/shell", "/dev/null" or "/bin/false"

```
nfsnobody:!!:11672:0:99999:7:::
nscd:!!:11672:0:99999:7:::
apache:!!:11672:0:99999:7:::
advu:$1$kcT7SU3v$LLOj/1DfwGq8/56f1k81./:11672:0:99999:7:::
bbuser:!!:11673:0:99999:7:::
postfix:!!:11676:0:99999:7:::
ntp:!!:11721:0:99999:7:::
pcap:!!:11799:0:99999:7:::
sshd:!!:11833:0:99999:7:::
dnscache:!!:11902:0:99999:7:::
dnslog:!!:11902:0:99999:7:::
```

Interesting enough, only the user "root" and "advu" possesses valid passwords, all other accounts have the string "!!" as their password hashes, which effectively prevents anyone from logging into those accounts. So much for our prior concerns over those interactive accounts "rpm" and "bbuser". While "advu" is the only other account with both a valid password hash and an interactive shell, its UID (i.e. user ID) of 500 suggests this is a non-root account. Nevertheless, let's investigate further on this.

To investigate the access logs on this system, the command "last" is used. The command "last" searches back through the file /var/log/wtmp (or the file designated by the -f flag) and displays a list of all users logged in (and out) since that file was created[5]. In this case, I directed the "last" command to consult the log file at /mnt/forensic/var/log/wtmp for its output.

```
[root@forensic forensic]# last -f var/log/wtmp
advu     pts/0        192.168.254.10   Wed Oct 16 00:51 - down   (00:10)
advu     pts/0        192.168.254.10   Wed Oct 16 00:32 - 00:33  (00:01)
advu     pts/0        192.168.254.11   Wed Oct 16 00:28 - 00:31  (00:03)
advu     pts/0        192.168.254.11   Wed Oct 16 00:19 - 00:23  (00:04)
reboot   system boot  2.4.9-34         Tue Oct 15 01:23         (23:38)
advu     pts/0        wolverine        Tue Oct 15 01:18 - down   (00:02)
advu     pts/0        wolverine        Mon Oct 14 22:30 - 22:30  (00:00)
advu     pts/0        wolverine        Mon Oct 14 14:47 - 15:36  (00:49)

wtmp begins Mon Oct 14 14:47:40 2002
```

It appears "advu" is the only user that has been access this system since October 14th, 2002. And that (s)he has been accessing this system from 192.168.254.11 and a host called "wolverine". Let's dig even deeper on this.

Since the current "wtmp" file began on Oct 14th 2002, a mere two days prior to the shutdown of the server for investigation. Perhaps we ought to look at any prior "wtmp" files for further clues. At the same time, I wonder what IP address was referred to by this host for the hostname "wolverine".

[5] Man page, July, 29th, 1999

```
[root@forensic forensic]# ls -l var/log/wtmp*
-rw-rw-r--   1 root    utmp       24960 Oct 16 01:02 var/log/wtmp
-rw-rw-r--   1 root    utmp        1536 Sep 17  2002 var/log/wtmp.1
[root@forensic forensic]# last -f var/log/wtmp.1
advu     pts/0        colossus       Tue Sep 17 16:03 - 16:04  (00:00)
advu     pts/0        colossus       Tue Sep 17 09:52 - 10:06  (00:14)

wtmp.1 begins Tue Sep 17 09:52:13 2002
[root@forensic forensic]# grep hosts etc/nsswitch.conf
hosts:      files nisplus dns
[root@forensic forensic]# cat etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1                localhost.localdomain localhost
192.168.254.102          storm storm.myhome.local
```

Upon reviewing the "wtmp.1" (the previous wtmp file), I discovered the user "advu" had also accessed this system from a host called "colossus". However, the "etc/hosts" file was not able to provide the information on the whereabouts of host "wolverine" and "colossus"… However, since the "wtmp" files contains only the short name of the hosts (instead of their fully qualified domain name), it is safe to say host "wolverine" and "colossus" are machines belonging to the same DNS domain and thus in the local network. Still no sign of connections from outside of the local network…

After examining both the list of all SUID/SGID binaries and the password files, I have concluded no apparent sign of backdoors on this system.

Signs of a Sniffer program

For a Sniffer program to be deployed by the intruder, at least some of the following needs to hold true:

1. Sniffer binary, either SUID/SGID or the intruder must have root privilege.
2. Data capture file, probably in a hidden directory
3. An automated mechanism to launch the Sniffer, possibly during boot-up

Hunting for Sniffer binary

As shown previously, none of the SUID/SGID files were neither foreign (based on their names) nor abnormal (based on their MD5 checksums, as verified by rpm). The only possibility here would be for the intruder to have somehow obtained the root privilege and was able to use the system's native Sniffer utility "tcpdump" for this purpose. Even if that were the case, there must be a file somewhere on the system being used to store all the data captured.

Hunting for data capture file

My first assumption would be a hidden directory. The "find" command is then again being used to locate and print out information about any "hidden" directories. This time,

we instruct the "find" command to locate any directories (with the "-type d" parameter) that starts with the character "." (with the "-name ".*" parameter) and display the result with the format of "%Tc %k %h/%f" followed by a new line character (i.e. \n). The formatting symbols are explained as follows:

- %f File's name with any leading directories removed (only the last element).
- %h Leading directories of file's name (all but the last element).
- %k File's size in 1K blocks (rounded up).
- %Tc File's last modification time in the format specified by c, which is locale's date and time (Sat Nov 04 12:02:33 EST 1989)

```
[root@forensic forensic]# find /mnt/forensic/ -name ".*" -type d -printf "%Tc %k %h/%f\n"
Thu 20 Dec 2001 08:39:00 PM EST 1 /mnt/forensic/root/.ssh
[root@forensic forensic]# ls -a /mnt/forensic/root/.ssh/
. .. known_hosts2
[root@forensic forensic]# cat /mnt/forensic/root/.ssh/known_hosts2
colossus,192.168.254.101 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAsGhUl0fBBMydJLHwdA7asarKYv0bZE8f2DRawaAnMi
dVd+gVV30TWs34y4dHBkC1gcTNWFWCcFJrpDXR1ZAbhoMRFcNiJc5sTMRwpHAOBnnr0prg
B3Tdit11OAgUvMIUeX2+tPmGlLNRJTkH2B/QXMV2o/+ZI1mRTP46DTpM4xE=
```

In my search for hidden directories, I found only one that seemed to created by the "ssh" (Secure Shell) application and contains only the "known_hosts2" file, which is a mere list of known hosts' RSA keys for Secure Shell. However, base on the info in this file, we were able to confirm our prior assumption of "colossus" being an internal host in the local network. Anyhow, let's see if we could find any "hidden" file in regular directories that might be used by the potential intruder for Sniffer data storage. Using the previous "find" command, I merely changed the file type to "directories" (i.e. with the option "-type f").

```
[root@forensic forensic]# find /mnt/forensic/ -name ".*" -type f -printf "%Tc %k %h/%f\n"
Sun 24 Feb 2002 12:58:11 AM EST 0 /mnt/forensic/var/spool/at/.SEQ
Mon 09 Jul 2001 08:56:20 AM EDT 1 /mnt/forensic/etc/skel/.bash_logout
Mon 09 Jul 2001 08:56:20 AM EDT 1 /mnt/forensic/etc/skel/.bash_profile
Mon 09 Jul 2001 08:56:20 AM EDT 1 /mnt/forensic/etc/skel/.bashrc
Sat 15 Dec 2001 07:27:08 PM EST 0 /mnt/forensic/etc/.pwd.lock
Wed 20 Feb 2002 03:06:55 PM EST 61 /mnt/forensic/usr/lib/perl5/5.6.1/i386-linux/.packlist
Mon 09 Jul 2001 08:56:19 AM EDT 1 /mnt/forensic/usr/share/man/man1/..1.gz
Sat 15 Dec 2001 08:03:33 PM EST 1 /mnt/forensic/home/advu/.bash_logout
Sat 15 Dec 2001 08:03:33 PM EST 1 /mnt/forensic/home/advu/.bash_profile
Sat 15 Dec 2001 08:03:33 PM EST 1 /mnt/forensic/home/advu/.bashrc
Wed 16 Oct 2002 01:02:41 AM EDT 6 /mnt/forensic/home/advu/.bash_history
Sun 16 Dec 2001 07:03:57 PM EST 1 /mnt/forensic/home/bbuser/.bash_logout
Sun 16 Dec 2001 07:03:57 PM EST 1 /mnt/forensic/home/bbuser/.bash_profile
Sun 16 Dec 2001 07:03:57 PM EST 1 /mnt/forensic/home/bbuser/.bashrc
Mon 20 May 2002 09:43:11 PM EDT 2 /mnt/forensic/home/bbuser/.bash_history
Wed 10 May 2000 11:51:05 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/install/.helloworld
```

```
Wed 10 May 2000 11:51:07 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/tmp/.helloworld
Mon 20 May 2002 08:52:18 PM EDT 1 /mnt/forensic/home/bbuser/bb19c/tmp/.license
Wed 16 Oct 2002 01:01:10 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/tmp/.expire
Wed 10 May 2000 11:51:08 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/ext/.helloworld
Tue 08 Aug 2000 01:44:46 PM EDT 0 /mnt/forensic/home/bbuser/bb19c/www/rep/.helloworld
Wed 10 May 2000 11:51:08 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/www/notes/.helloworld
Wed 10 May 2000 11:51:12 AM EDT 0 /mnt/forensic/home/bbuser/bb19c/www/html/.helloworld
Wed 10 May 2000 11:51:08 AM EDT 0 /mnt/forensic/home/bbuser/bb18d4/www/notes/.helloworld
Tue 08 Aug 2000 01:44:46 PM EDT 0 /mnt/forensic/home/bbuser/bb18d4/www/rep/.helloworld
Wed 10 May 2000 11:51:12 AM EDT 0 /mnt/forensic/home/bbuser/bb18d4/www/html/.helloworld
Wed 10 May 2000 11:51:05 AM EDT 0 /mnt/forensic/home/bbuser/bb18d4/install/.helloworld
Wed 10 May 2000 11:51:07 AM EDT 0 /mnt/forensic/home/bbuser/bb18d4/tmp/.helloworld
Wed 16 Jan 2002 11:44:46 PM EST 1 /mnt/forensic/home/bbuser/bb18d4/tmp/.license
Mon 20 May 2002 09:41:03 PM EDT 0 /mnt/forensic/home/bbuser/bb18d4/tmp/.expire
Wed 10 May 2000 11:51:08 AM EDT 0 /mnt/forensic/home/bbuser/bb18d4/ext/.helloworld
Thu 20 Dec 2001 06:08:18 PM EST 1 /mnt/forensic/home/postfix/.bash_logout
Thu 20 Dec 2001 06:08:18 PM EST 1 /mnt/forensic/home/postfix/.bash_profile
Thu 20 Dec 2001 06:08:18 PM EST 1 /mnt/forensic/home/postfix/.bashrc
Sat 10 Jun 2000 05:00:15 PM EDT 1 /mnt/forensic/root/.bash_logout
Wed 23 Aug 1995 03:02:38 PM EDT 2 /mnt/forensic/root/.Xresources
```

Once again, nothing interesting showed up here. Almost all hidden files found bare common names and are too small to be Sniffer data files. I verified the larger ones (e.g. the .bash_history files and the .packlist file) and concluded their content to be legitimate.

Another thought, maybe this data capture file is kept at the "/dev" directory; a common place for intruder to hide files in. Now, all files in the "/dev" directory are either a character-mode device or a block-mode device, so files that does not possess either one of these attributes would be suspicious to us. As usual, the "find" command is being utilized to locate the suspicious files. Given the nature of our search (i.e. looking for files that are neither block-mode nor character-mode) and the desired output format (like the previous search), the option "-type f -not –type c –not –type b –printf "%Tc %k %h/%f\n" is once again used.

```
[root@forensic forensic]# find /mnt/forensic/dev -type f -not -type b -not -type c -printf "%Tc %k
%h/%f\n"
Thu 30 Aug 2001 04:30:52 PM EDT 18 /mnt/forensic/dev/MAKEDEV
```

As it turns out, we did not find any suspicious files in the "/dev" directory, either. The only non-character/block mode file is the MAKEDEV executable, which was verified previously by the "rpm –root /mnt/forensic/ -Va" command. Base on the data gathered thus far, I drew another blank on this one.

Hunting for the automated Sniffer-launching mechanism

In order for a program to be executed automatically, it would have to be invoked directly or indirectly through either a system start scripts or through a task-scheduler. Let's focus on the task-scheduler for now, as I will investigate the system start scripts in a later section.

In Linux systems, "at" and "crontab" are two readily available task schedulers. The "at" scheduler read its task schedules from the directory "/var/spool/at/spool" while the "crontab" scheduler reads its task schedules from the directories:

- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly
- /etc/cron.d
- /var/spool/cron

I then proceeded to check each of those directories. For the most part, all task schedules in those directories were placed by their RPMs. We determined these relationships by looking for (using the good old "grep" command) each file in the list the files queried from the RPM database for the appropriate package (using the "rpm –ql PackageName" command).

```
[root@forensic etc]# pwd
/mnt/forensic/etc
[root@forensic etc]# ls cron.hourly/
```

Nothing in the /etc/cron.hourly directory

```
[root@forensic etc]# ls cron.daily/
00webalizer  logrotate       rpm          sysstat
0anacron     makewhatis.cron  slocate.cron  tmpwatch
[root@forensic etc]# rpm --root /mnt/forensic/ -ql webalizer | grep /etc/cron.daily
/etc/cron.daily/00webalizer
[root@forensic etc]# rpm --root /mnt/forensic/ -ql anacron | grep /etc/cron.daily
/etc/cron.daily/0anacron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql logrotate | grep /etc/cron.daily
/etc/cron.daily/logrotate
 [root@forensic etc]# rpm --root /mnt/forensic/ -ql man | grep /etc/cron.daily
/etc/cron.daily/makewhatis.cron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql rpm | grep /etc/cron.daily
/etc/cron.daily/rpm
 [root@forensic etc]# rpm --root /mnt/forensic/ -ql slocate | grep /etc/cron.daily
/etc/cron.daily/slocate.cron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql sysstat | grep /etc/cron.daily
/etc/cron.daily/sysstat
[root@forensic etc]# rpm --root /mnt/forensic/ -ql tmpwatch | grep /etc/cron.daily
/etc/cron.daily/tmpwatch
```

Each of files found in /etc/cron.daily directory was placed there by their RPMs

```
[root@forensic etc]# ls cron.weekly/
0anacron  makewhatis.cron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql anacron | grep /etc/cron.weekly
/etc/cron.weekly/0anacron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql man | grep /etc/cron.weekly
/etc/cron.weekly/makewhatis.cron
[root@forensic etc]# ls cron.monthly/
0anacron
[root@forensic etc]# rpm --root /mnt/forensic/ -ql anacron | grep /etc/cron.monthly
/etc/cron.monthly/0anacron
[root@forensic etc]# ls cron.d
sysstat
[root@forensic etc]# rpm --root /mnt/forensic/ -ql sysstat | grep /etc/cron.d/sysstat
/etc/cron.d/sysstat
[root@forensic spool]# pwd
/mnt/forensic/var/spool
[root@forensic spool]# ls cron/
root
[root@forensic spool]# cat cron/root
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.2973 installed on Wed Aug 14 14:13:35 2002)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
0 * * * * /usr/local/etc/logcheck.sh
0 12 * * * /root/share/update.cron
[root@forensic spool]# cat /mnt/forensic/root/share
cat: /mnt/forensic/root/share: No such file or directory
[root@forensic spool]# ls -l /mnt/forensic/root/
total 2
-rw-r--r--    1 root     root         1089 Dec 15  2001 anaconda-ks.cfg
lrwxrwxrwx    1 root     root           17 Feb  3  2002 share -> /mnt/import/root/
[root@forensic spool]# ls -l /mnt/forensic/mnt/import/
total 0
[root@forensic spool]# grep /mnt/import /mnt/forensic/etc/fstab
//cyclops/pub                   /mnt/import                   smbfs
        username=storm,password=storm,fmask=744,rw
[root@forensic spool]# ls at/spool
[root@forensic spool]#
```

Each of files found in /etc/cron.weekly directory was placed there by their RPMs

The files found in /etc/cron.monthly directory was placed there by its RPM

The file found in /etc/cron.d directory was placed there by its RPM

What are these commands?

Nothing is found at the /var/spool/at/spool directory

At the same time, however, we noticed two commands being scheduled from the root user's personal crontab file "/var/spool/cron/root":

- /usr/local/etc/logcheck.sh
- /root/share/update.cron

After close examination, "logcheck.sh" seems to belong to Psionic Technologies' LogSentry product. Unfortunately, this product is no longer available after Cisco Systems acquired the firm. Nevertheless, this bourne shell script does not seem to carry out any suspicious activities, after a careful visual inspection of its code.

However, I was not able to locate the "update.cron" file as it was stored on a remote SMB file system, as determined by the entry in the system's "/etc/fstab" file.

At the conclusion of my search for Sniffer on this system, I was not able to conclude the presence of such binary nor its data capture storage on this system.

History files

One excellent feature about Linux systems is its logging capabilities. All of the common shells (or command interpretation systems) by default keep a good record of each command issued by the user. In this section, I will identify and examine each of these command history files.

By default, those command history files are written to a hidden file on the user's home directory. As shown from our previous search for hidden files on this system, we have located the following command history files:

1. /home/advu/.bash_history
2. /home/bbuser/.bash_history
3. /root/.bash_history

Each of the above files were carefully examined but could not locate any suspicious activities in them. The activities of interests are basically any outbound network connectivity and program compilations.

I located only regular telnet, ftp and ssh connections to various internal hosts and to two external hosts, both of which are legitimate business partner of the school board. The high school's system administrator was also able to identify those outbound connections as part of his routine functions.

Additionally, all occurrence of program compilation (e.g. the commands "make" or "gcc") were traced back to either the upgrade of the "Big Brother" network monitoring system or the upgrade of the "postfix" mail transfer agent. The first case was easy, due to the use of the account name and its uniquely cascaded commands. In the case of the program compilations in the root user's history file, the system administrator was able to identify the pattern of "make" and "make upgrade" pairs of commands for me.

```
[root@forensic forensic]# pwd
/mnt/forensic
[root@forensic forensic]# grep make home/advu/.bash_history
[root@forensic forensic]# grep gcc home/advu/.bash_history
[root@forensic forensic]# grep make home/bbuser/.bash_history
make;make install;cd ../..;chown -R bbuser bb*
make;make install;cd ../..;chown -R bbuser bb*
cd ../src;make;make install; cd ../..; chown -R bbuser:bbuser bb19c
[root@forensic forensic]# grep gcc home/bbuser/.bash_history
[root@forensic forensic]# grep make root/.bash_history
make; make upgrade
make
make upgrade
make
make upgrade
make
make upgrade
make
make upgrade
make
make setup check
make linux
[root@forensic forensic]# grep gcc root/.bash_history
[root@forensic forensic]#
```

Big Brother upgrades

Postfix upgrades, as identified by the system administrator

In conclusion of this section, we found no evidence of suspicious activities.

/proc examination

Oh well, it appeared to me that this computer was properly shut down. As the result, there is no data in the "/proc" directory. One must make sure the system administrator do not do this in the future when being asked to submit hard drives for forensic analysis.

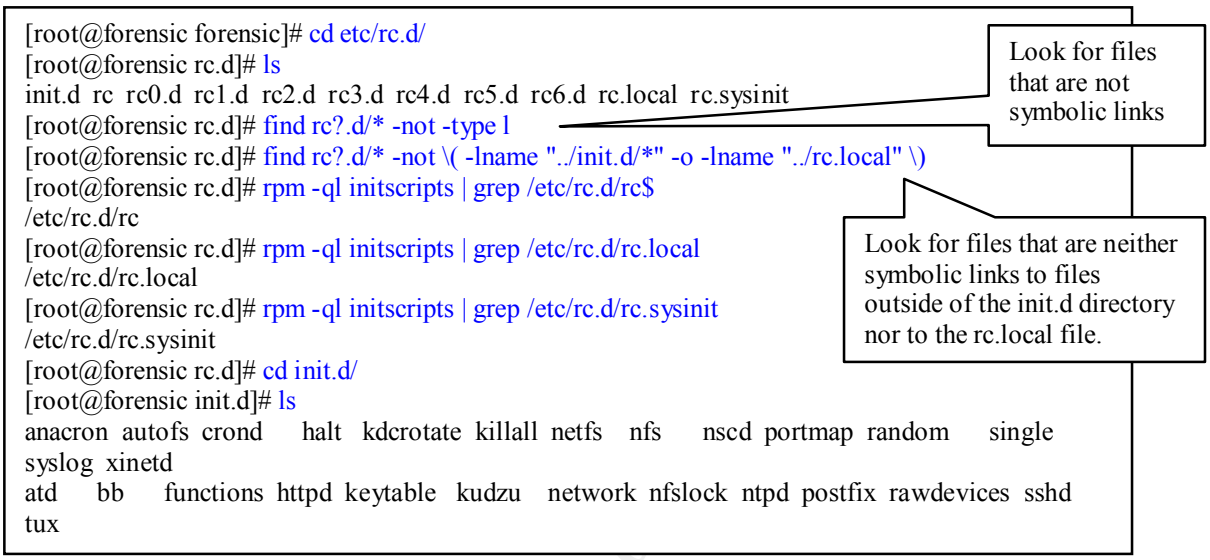
```
[root@forensic forensic]# cd proc
[root@forensic proc]# ls -ld .
drwxr-xr-x    2 root     root         1024 Dec 15  2001 .
```

Start up files

Finally, I will examine the system start up files to attempt locating traces or clues to the potential security breach on this host.

In a RedHat Linux system, the system start up scripts are kept at the directory "/etc/rc.d". Typically, one will find three files (rc, rc.local and rc.sysinit) and 8 subdirectories (init.d, rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d, and rc6.d) under this directory. Most of the time, subdirectories "rc0.d" through "rc6.d" contains merely symbolic links to actual start up scripts in the "init.d" subdirectory or symbolic links to the rc.local script.
To help make short of verifying the content of the rcX.d directories, I first search for non symbolic links in those directories (using the "find rc?.d/* -not –type l" command, where the wildcard "rc?.d/*" resolves to all files in those rcX.d directories and the parameter "-

not –type l" instructs the "find" command to search for non symbolic link files) and then validate those symbolic references using the command "find rc?.d/* -not \(-lname "../init.d/*" –o –lname "../rc.local" \)". The "-lname" parameter specifies the symbolic link pattern for "find" to match against.



Through detail examination of the start up scripts in the "/etc/rc.d/init.d" directory (via a series of queries to the RPM database), I was able to verify the integrity of all but two start up scripts: "bb" and "postfix". However, by simply walking through the content of these scripts, I was able to ascertain their uses are nothing more than to merely launch the "BigBrother" and the "Postfix" daemons respectively.

In summary, I found no evidence through the analysis of the start up scripts to support the occurrence of any tempering.

Non-modification of the evidence during the analysis

Finally, MD5 checksum is taken once again on the image file "hdb1image.img" and compared to the checksum we obtained previously. By the virtue of successful MD5 checksum verification, we are certain the image file (i.e. the evidence) was not modified during this examination.

Timeline Analysis

Timeline analysis is often valuable to an investigation. It provides the investigator with a detailed list of when each file was last modified, created and accessed. If the time of the compromise were known, we could then focus our effort looking at changes/events taking place around that timeframe. However, as in this case, the time of the compromise is not known to us.

Obviously, this provides the forensic investigator with a quick and easy way to find the relationship between the creation and the modification of various files and directories. Rather often, we are able to reconstruct the activity of the intruder by simply follow the timeline around the creation date of suspicious files or directories.

Additionally, from this summary of times, we are able to gather key information about the system such as:

- System installation time
  The installation time of the system can often be traced to when a large amount of files and directories were created (i.e. the cTime being changed on a high number of files and directories).

- Time of major updates
  Similarly, a major system update would present itself as large number of consecutive file creations after the initial system installation.

- Time of last use
  Obviously, this is simply the date of the last entry in the timeline file.

- Time of last software compilation
  Apart from the obvious use of the *cc compilers, a high number of library files (i.e. *.h files) will be indicative of when source codes were compiled on the system.

From the forensic workstation, I added the following entry to the "/morgue/fsmorgue" file:

```
hdb1image.img   linux-ext2      /       EST5EDT
```

Then the following commands were executed to launch the "Autopsy Forensic Browser"[6].

[6] A browser interface to The @stake Sleuth Kit (TASK), created by Brian Carrier of @stake

```
[root@forensic morgue]# cd /usr/local/autopsy
[root@forensic autopsy]# ./autopsy 8888 localhost &
[1] 7241
[root@forensic autopsy]#
================================================================================

                    Autopsy Forensic Browser
                        ver 1.50

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

Morgue: /morgue
Start Time: Fri Mar 21 11:39:04 2003
Investigator: Christopher Lee

Paste this as your browser URL on localhost:
        localhost:8888/13559782742903294675/autopsy

Press ctrl-c to exit

[root@forensic autopsy]# netscape localhost:8888/13559782742903294675/autopsy &
[2] 7242
```

From the browser, a forensic data file "body" was created for "hdb1image.img" with the options to collect data of types "Allocated Files", "Unallocated Files" and "Unallocated Inodes".

Once the "body" file was created, a timeline file "timeline" is then created with no explicit start nor end dates (i.e. all dates). This "timeline" file documents the changes in the MAC time of each file (both allocated and unallocated files, including the unallocated Inodes). In this case, this file is 9,879,218 bytes in size and contains 94841 lines. This is far too large to examine manually.

```
[root@forensic morgue]# ls -l timeline
-rw-r--r--   1 root     root      9879218 Mar 21 12:19 timeline
[root@forensic morgue]# wc -l timeline
  94841 timeline
```

To facilitate the analysis of this rather large timeline file, I wrote a BASH (Bourne-Again SHell) script to carry out the following:

1. Ensure the time stamp is printed in front of each entry within the timeline file and save the new timeline file as "timeline.complete".
2. Summarize the timeline files by highlighting the number of file/directory creations on a ten-minute interval. This summary is stored in the "timeline.creation" file.

```
#!/bin/sh

# First, ensure the date is printed in front of every line
C=1
IFS=

read LINE
while [ $? -eq 0 ]
do
        F1=`echo "$LINE" | cut -c1-20`
        if [ "$F1" == "                    " ]; then
                F1=$OLDF1
        fi
        F2=`echo "$LINE" | cut -c21-`
        echo "$F1" "$F2" >> timeline.complete
        (( C=$C + 1 ))
        OLDF1=$F1
             read LINE
done

# summarize the timeline for creation of files/directories
grep '[m.][a.]c' timeline.complete | cut -c1-16 | sort -b -k 4.3 | sort -b -k 4.1,4.2 | sort -b -n -k 2.1,2.2 |
sort -M -b -k 1.,1.3 | sort -b -k 3.1,3.4 | uniq -c | awk -F':' '{print $1":"$2"0-"substr($1,21,22)":"$2"9"}'
> timeline.creation
```

Summary file is generated by one continuous command.

By analyzing the "timeline.creation" file, I was able to ascertain the system installation time to be December 15th, 2001, between 7pm and 8:19pm. Similarly, I was also able to identify major system upgrade (or major additional software installation) took place between 11:30-12:10pm on Aug 14th, 2002 (10,300+ files were created).

```
[root@ forensic root]# more timeline.creation
     1 Dec 31 1969 19:00-19:09
     1 Dec 31 1969 19:10-19:19
……..
   316 Dec 15 2001 19:10-19:19
 10552 Dec 15 2001 19:20-19:29
  7120 Dec 15 2001 19:30-19:39
  1004 Dec 15 2001 19:40-19:49
   786 Dec 15 2001 19:50-19:59
    26 Dec 15 2001 20:00-20:09
```

Entries were ignored, since they predate the installation date. Those are files copied to the system through the RPM packages.

Simply by examining the end of the "timeline.complete" file, I was able to determine this system was last used (shut down by the system administrator) at 1:02am on Oct 16th, 2002.

```
[root@forensic root]# tail timeline.complete
Oct 16 2002 01:02:48       0 mac ---------- 0       0       26358    <hdb1image.img-dead-26358>
Oct 16 2002 01:02:48       0 ..c -/-rw-r--r-- 0       0       12       /.autofsck (deleted)
Oct 16 2002 01:02:48       0 ..c -rw-r--r-- 0       0       12       <hdb1image.img-dead-12>
```

```
Oct 16 2002 01:02:48      0 mac -/-rw-r--r-- 0      0      26487    /etc/.webalizer.conf.swx (deleted)
Oct 16 2002 01:02:48  28380 .a. -/-rwsr-xr-x 0      0      84860    /bin/umount
Oct 16 2002 01:02:48      0 mac -rw-r--r-- 0      0      26487    <hdb1image.img-dead-26487>
Oct 16 2002 01:02:48     13 .a. l/lrwxrwxrwx 0      0      8095     /lib/libc.so.6 -> libc-2.2.4.so
Oct 16 2002 01:02:48      0 mac -/---------- 0      0      26358    /etc/.webalizer.conf.swp (deleted)
Oct 16 2002 01:02:48  11363 .a. -/-rw-r--r-- 0      0      26452    /etc/ld.so.cache
Oct 16 2002 01:02:48     47 mac -/-rw-r--r-- 0      0      26852    /etc/mtab
```

Further, based on the usage of the C library files (*.h files), I was able to make an educated guess that some source code was compiled on Jan 8th 2002, Jan 16th 2002, Feb 8th 2002, Feb 15th 2002, Feb 20th 2002, May 20th 2002, Jun 1st 2002, Jul 3rd 2002, Jul 10th 2002, Aug 3rd 2002, Aug 7th 2002 and Aug 14th 2002.

```
[root@forensic root]# grep "[m.]a\." timeline.complete | grep "\.h$" | cut -c1-11 | uniq -c
……..
      2 Jan 08 2002
      1 Jan 16 2002
      2 Feb 08 2002
      1 Feb 15 2002
     47 Feb 20 2002
      1 May 20 2002
    420 Jun 01 2002
      7 Jul 03 2002
      1 Jul 10 2002
     72 Aug 03 2002
    305 Aug 07 2002
     39 Aug 14 2002
```

Entries were ignored, since they predate the system installation date.

While none of these dates fall within close proximity of the intrusion to the database server, the possibility of this host being compromised much earlier still remains. However, without further evidence to support this claim or to suggest a time frame for which this host might have been compromised, I am not able conclude any misuse did take place on this host.

Recover Deleted Files

Often time, the intruder will delete either the source code of their rootkit or the package containing the rest of his tools once he gain access to the compromised system and accomplished his objective. By recovering deleted files, the investigator can often gain further insight or evidence of the intruder's activities on this system.

To recover the deleted files on this system, the following piece of code was created to automat the functionality of tools from "The @stake Sleuth Kit" (TASK)[7]. In short, this sequence of commands parses the output of the "ils" command from "TASK", and provides it as the input to the "icat" command (from "TASK" as well) to recover data contained in those inodes. The recovered deleted files are place in the "/morgue/deleted" directory.

The "ils" command displays the information on the specified inode. The "-r" parameter instructs the program to look for only inodes of "deleted" files, while the "-f" parameter specifies the file system of the device/image file to be investigated.

The "awk" processor is then applied to parse and format the output of the "ils" command to proper input strings for the subprogram following.

Finally, within the "while" loop, the "icat" command extracts the data in each of the inodes of deleted files and save it into files in the "/morgue/deleted" directory, named after their respective inode numbers.

```
[root@forensic bin]# mkdir /morgue/deleted
[root@forensic bin]# cd /usr/local/task/bin
[root@forensic bin]# ./ils -r -f linux-ext2 /morgue/hdb1image.img | awk -F '|' '($2=="f") {print $1}' |
while read i; do ./icat -f linux-ext2 /morgue/hdb1image.img $i > /morgue/deleted/$i; done
[root@forensic bin]#
```

As the result of the data recovery, 6,978 files were created in the directory "/morgue/deleted", however the majority of them are 0 bytes (i.e. those deleted inodes have since been reallocated to other files). Even the non-zero byte files possess no more than 46 bytes of data and of which are entirely binary in nature. Clearly, none of the deleted files are of interests to the investigation of this system.

String Search

In principle, the forensic media should be searched for files containing keywords that we have picked up during the investigation thus far. This technique is often employed to locate additional suspected files altered/left behind by the intruder.

In addition to the above "clues", the following list of keywords has often been found within blackhat tools "r00tkit, rootkit, hack, irc, bot, sniff, backdoor, promisc, knark, hax0r, hide, Trojan, virus, TFN2K, adore, LKM, attack, denial –of- service, ddos, brute force, 0wn". These keywords are often part of the blackhat's binaries' embedded text.

[7] The @stake Sleuth Kit (TASK) is the only open source forensic toolkit for analyzing Microsoft and UNIX file systems. It integrates the file system analysis tools of The Coroner's Toolkit (TCT), by Wietse Venema and Dan Farmer, with TCTUTILs and adds new features. TASK is written by Brian Carrier of @stake.

In this case, given that we have not been able to locate any clue of intrusion/misuse up to this point. I proceed with a case insensitive keyword search on the list of common blackhat keywords documented in the previous paragraph.

"Autopsy Forensic Browser" searches the raw device one keyword at a time and returns a list of inodes (both allocated and unallocated) containing the matched keyword. While it's easy to use, it can be time-consuming and cumbersome for interpretation of the data. Since we have already salvaged the deleted files in the previous section, an (better) alternative strategy is employed here to look for files containing those keywords of interests:

1. Enter those keywords into the "blackhat_strings", where each keyword, prefixed and suffixed with a blank space, is at its own line.
2. Create a executable shell script "find_keywords" as follows:

   ```
   #!/bin/sh
   # this scripts prints out the filenames of files that contains strings
   # in the blackhat_strings file.
   fgrep -f /morgue/blackhat_strings $1 > /dev/null && echo $1
   ```

3. Execute the command:

   "find /mnt/forensic/ -type f -exec ./find_keywords {} \; | cut -c14- | while read i ; do rpm -V –nodeps --root=/mnt/forensic/ -f $i | grep $i ; done"

   The result is a list of either suspicious regular files containing one or more of these keywords and the description on where they failed the verification process.

4. Execute the command:

   "find /morgue/deleted/ -type f –exec ./find_keywords {} \;"

   The result is a list of deleted files containing one or more of these keywords.

The result of this search is reveals the following list of "suspicious files":

1. /etc/postfix/canonical
2. /usr/local/etc/logcheck.sh
3. /usr/local/man/man5/canonical.5
4. /usr/local/man/man8/cleanup.8
5. /home/bbuser/bb19c/web/bb-ack.sh.DIST
6. /home/bbuser/bb19c/web/bb-hist.sh.DIST
7. /home/bbuser/bb19c/www/help/bb-man.html
8. /home/bbuser/bb18d4/www/help/bb-man.html
9. /home/bbuser/bb18d4web/bb-ack.sh.DIST
10. /home/bbuser/bb18d4web/bb-hist.sh.DIST
11. /var/cache/man/whatis
12. /etc/webalizer.conf

File 12 was quickly verified as the legitimate configuration file for "Webalizer" (a website usage reporting tool), while files 5 through 10 were quickly identified as the shell script templates supplied with version 19c and version 18d4 of the BigBrother network monitoring system. A closer look confirmed files 1 through 4 are legitimate files for the

```
[root@forensic morgue]# find /mnt/forensic/ -type f -exec ./find_keywords {} \; | cut -c14- | while read
i ; do rpm -V –nodeps --root=/mnt/forensic/ -f $i  | grep $i ; done
file /var/cache/man/whatis is not owned by any package
error: file /etc/postfix/canonical: No such file or directory
S.5....T c /etc/webalizer.conf
error: file /usr/local/etc/logcheck.sh: No such file or directory
error: file /usr/local/man/man5/canonical.5: No such file or directory
error: file /usr/local/man/man8/cleanup.8: No such file or directory
error: file /home/bbuser/bb19c/web/bb-ack.sh.DIST: No such file or directory
error: file /home/bbuser/bb19c/web/bb-hist.sh.DIST: No such file or directory
error: file /home/bbuser/bb19c/www/help/bb-man.html: No such file or directory
error: file /home/bbuser/bb18d4/www/help/bb-man.html: No such file or directory
error: file /home/bbuser/bb18d4/web/bb-hist.sh.DIST: No such file or directory
error: file /home/bbuser/bb18d4/web/bb-ack.sh.DIST: No such file or directory
```

"Postfix"[8] mail transfer agent software, which was compiled and installed manually onto this host.

File 11 (i.e. /var/cache/man/whatis) is the keyword index of all man pages and a quick test revealed its innocence.

```
[root@forensic forensic]# fgrep -f /morgue/blackhat_strings ./var/cache/man/whatis
perlhack          (1)  - How to hack at the Perl internals
perltie           (1)  - how to hide an object class in a simple variable
shred             (1)  - delete a file securely, first overwriting it to hide its contents
```

Okay, so we have nothing in the ext3 file system that interests us. Let move onto the swap partition and find out if we have better luck there (e.g. in case the program were launched from a remotely mounted file system).

```
[root@forensic morgue]# ./find_keyword hdb2image.img
[root@forensic morgue]#
```

Unfortunately (for the investigation), the keyword search did not turn up any useful clue on this system.

Conclusions

[8] A clean, efficient, and secured MTA written by Wietse Venema

Based on the result of all of the above analysis, I have yet to locate concrete evidence of misuse on this system. While it is possible for the intruder to have covered his tracks so carefully that he leaves no visible forensic data behind, it would have required in-depth knowledge and expertise typically not found on people who are interested in the database server of a local high school.

Therefore, after thoroughly analyze the file system, the timeline file, the deleted files, and even the swap file, I am confident to conclude that it is highly unlikely for this system to have been compromised by an external intruder and used as a launch pad to attack the internal infrastructure at the local high school.

Part 3 - Legal Issues of Incident Handling

Background:

I am the system administrator for a regional Internet Service Provider that provides Internet access to paying customers. I receive a telephone call from a law enforcement officer who informs you that an account on my system was used to hack into a government computer. He asks me to verify the activity by reviewing my logs and determine if my logs reflect whether or not the activity was initiated there or from another upstream provider. I review my 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 purposes of this scenario, assume I validated the identity of the law enforcement officer and this is not social engineering.

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

This question can be best answered by addressing concerns over the following three areas:

1. Compliance with corporate policy
2. Compliance with government privacy legislature
3. Compliance with government criminal code

Compliance with Corporate Policy

While the corporate policy of each ISP is different, the majority of Canadian ISPs belongs and subscribed to the same Code of Conduct and Privacy Code standards as established by the members of the CAIP (Canadian Association of Internet Providers).

While I, as a system administrator of an ISP, is obliged to assist the law enforcement officers (section 1, Code of Conduct), necessary authorization or user consent still must be received prior to the release of the information. In this case, the nature of a criminal investigation satisfies the provision for exemptions of the principle 3 of the Privacy Code.

At the same time, the disclosure of the information must also comply with the principle 5 of the Privacy Code, which states the release of information when required to do so by law (e.g. subpoenas, search warrants, other court and government orders, or demands from other parties who have a legal right to personal information, or to protect the security and integrity of its network or system) must be limited to the information that is legally required and nothing more.

While compliance to CAIP's codes is voluntary, it is typical of a service organization such as an ISP to try protecting its image as a trusted custodian of its users' private information. As the result, it is highly unlikely for an ISP to release private user

information without appropriate paperwork and legal requirements first being satisfied. For the sake of this hypothetical discussion, it is prudent to assume the prevalence of this conservative nature.

Therefore, based on the above codes, I am not able to release user information (including the log data) to the law enforcement officers over the phone without first receiving the appropriate paperwork.

The current CAIP's Code of Conduct is published at http://www.caip.ca/issues/selfreg/code-of-conduct/code.htm)

The current CAIP's Privacy Code is published at http://www.caip.ca/issues/selfreg/privacy-code/privacy.htm)

Compliance with Government Privacy Legislature:

Currently, in Canada, the government does not have any legislature outlining the proper handling of personal information for non-government institutions.

However, the federal Privacy Act (R.S. 1985, c. P-21) describes what "Federal Institutions" must adhere to for the collection, disclosure, protection, retention, and storage of personal information.

Similarly, the Ontario provincial government's "Freedom of Information and Protection of Privacy Act" describes the policy each provincial government institution must comply with in regards to the handling of personal information.

However, since I am the system administrator of a regional ISP that does not fall into the categories of federal or provincial institutions, there is no restriction in the disclosure of the log data. In case, being a good citizen, I will provide to the officer as much info over the phone as he requests.

References:

"Ontario Freedom of Information and Protection of Privacy Act", (http://www.ipc.on.ca/scripts/index_.asp?action=31&P_ID=11607&N_ID=1&PT_ID=23&U_ID=0#top)
"Canadian Federal Privacy Act", (http://laws.justice.gc.ca/en/P-21/index.html)

Compliance with Government Criminal Code:

Under Criminal Code section 487 (2.2), I will be required of full cooperation with the law enforcement officer when a warrant is presented.

However, in this case, a warrant has not been produced and as the result, I am not required under the law to provide such information. At the same token, since I am not required by the law to release the information, I am still bounded by the typical corporate policy of upholding the privacy of the requested user information.

Canadian Criminal Code Section 487 is available at http://www.canlii.org/ca/sta/c-46/sec487.html

Question 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?

Currently, there are no provisions under the existing Canadian legislature on the topics of "data-preservation" when a warrant cannot be obtained in a timely manner. As the result, the preservation of any "evidence" in the absence of a signed warrant is not required of any service providers.

Without the support of applicable legislature, I, as an administrator for an ISP, am not allowed to access potential evidences protected under the privacy legislatures. As such, nothing can be done to preserve the evidence without a signed warrant.

A proposed "Lawful Access" legislature (available through Canadian Justice Department website @ http://www.canada.justice.gc.ca/en/cons/la_al/) has been made available for public consultation since August 2002. However, service providers (such as ISPs) are fearful the cost of implementing and maintaining necessary infrastructure to support requirements proposed within this legislature will spell the doom of the industry, especially in today's economy condition.

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

As suggested by the section 487.01 (1) of the Canadian Criminal Code, a signed warrant is necessary to authorize a peace officer to request the release of those log files.

Such warrant will need to be signed by either a:

- provincial court judge,
- judge of a superior court of criminal jurisdiction, or
- judge defined in section 552 of Canadian Criminal Code.

Under section 552 of Canadian Criminal Code, a judge is defined as:

a) in the Province of Ontario, a judge of the superior court of criminal jurisdiction of the Province,
b) in the Province of Quebec, a judge of the Court of Quebec,
c) in the Province of Nova Scotia, a judge of the superior court of criminal jurisdiction of the Province,
d) in the Province of New Brunswick, a judge of the Court of Queen's Bench,
e) in the Province of British Columbia, the Chief Justice or a puisne judge of the Supreme Court,
f) in the Provinces of Prince Edward Island and Newfoundland, a judge of the Supreme Court,
g) in the Province of Manitoba, the Chief Justice or a puisne judge of the Court of Queen's Bench,
h) in the Provinces of Saskatchewan and Alberta, a judge of the superior court of criminal jurisdiction of the province,
i) in the Yukon Territory and the Northwest Territories, a judge of the Supreme Court of the territory, and
j) in Nunavut, a judge of the Nunavut Court of Justice;

References:

Criminal Code section 487.01 (1), http://www.canlii.org/ca/sta/c-46/sec487.01.html
Criminal Code section 552, http://www.canlii.org/ca/sta/c-46/sec552.html

Question D: What other "investigative" activity are you permitted to conduct at this time?

While there are no laws prohibiting me from "investigate" the crime, I must do so in compliance with all necessary privacy and criminal laws.

As the result, it is typically a "safe" practice to allow the law enforcement agents to drive the investigation instead of allowing my own zeal to thwart their efforts.

That being said, non-intrusive reviews of the organization's security strategy and/or vulnerabilities of its infrastructure are good example of things I could investigate and perhaps could bring about clues on how the intrusion might have had taken place. It is vitally important to ensure my own investigation is not interfering nor damaging the official investigation as well its compliance with appropriate laws and regulations.

Question 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?

If internal staff members of the ISP made this discovery of unauthorized access via a lawful process (such as a routine system maintenance or internal system audits), we could then gather the evidence and submit it, in its relevant entirety, to the appropriate law enforcement agency (in this case, Royal Canadian Mounted Police) as a report of violation of Criminal Code section 342.1[9] (Unauthorized Use of Computer, Unauthorized Possession of Password) and possibly section 342.2(1)[10] (Possession of Device to Obtain Computer Service).

However, if the law enforcement agency made the discovery of the unauthorized access, then the response procedure would have been consistent with what I have described in previous sections.

Once a formal criminal investigate has been taken place and I, as the ISP's system administrator, has been contacted by the law enforcement officers to cooperate with them, I could no longer "investigate" the crime on my own without proper authorization (typically in writing) from the appropriate law enforcement agencies.

[9] Criminal Code 342.1, http://www.canlii.org/ca/sta/c-46/sec342.1.html
[10] Criminal Code 342.2 (1), http://www.canlii.org/ca/sta/c-46/sec342.2.html

Appendix (for Part 1):

"trace-atd" file

(output of strace on the unknown binary)

```
08:55:24 execve("./atd", ["./atd"], [/* 16 vars */]) = 0
08:55:24 old_mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40007000
08:55:24 mprotect(0x40000000, 21772, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
08:55:24 mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
08:55:24 stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=10994, ...}) = 0
08:55:24 open("/etc/ld.so.cache", O_RDONLY) = 3
08:55:24 old_mmap(NULL, 10994, PROT_READ, MAP_SHARED, 3, 0) = 0x40008000
08:55:24 close(3)                       = 0
08:55:24 stat("/etc/ld.so.preload", 0xbffffba0) = -1 ENOENT (No such file or directory)
08:55:24 open("/usr/i486-linux-libc5/lib/libc.so.5", O_RDONLY) = 3
08:55:24 read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0(k\1\000"..., 4096) = 4096
08:55:24 old_mmap(NULL, 823296, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x4000b000
08:55:24 old_mmap(0x4000b000, 592037, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED,
3, 0) = 0x4000b000
08:55:24 old_mmap(0x4009c000, 23728, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED,
3, 0x90000) = 0x4009c000
08:55:24 old_mmap(0x400a2000, 201876, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a2000
08:55:24 close(3)                       = 0
08:55:24 mprotect(0x4000b000, 592037, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
08:55:24 munmap(0x40008000, 10994)      = 0
08:55:24 mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
08:55:24 mprotect(0x4000b000, 592037, PROT_READ|PROT_EXEC) = 0
08:55:24 mprotect(0x40000000, 21772, PROT_READ|PROT_EXEC) = 0
08:55:24 personality(0 /* PER_??? */)   = 0
08:55:24 geteuid()                      = 0
08:55:24 getuid()                       = 0
08:55:24 getgid()                       = 0
08:55:24 getegid()                      = 0
08:55:24 geteuid()                      = 0
08:55:24 getuid()                       = 0
08:55:24 brk(0x804c820)                 = 0x804c820
08:55:24 brk(0x804d000)                 = 0x804d000
08:55:24 open("/usr/share/locale/en_US.iso885915/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No
such file or directory)
08:55:24 stat("/etc/locale/C/libc.cat", 0xbffff6c4) = -1 ENOENT (No such file or directory)
08:55:24 stat("/usr/lib/locale/C/libc.cat", 0xbffff6c4) = -1 ENOENT (No such file or directory)
08:55:24 stat("/usr/lib/locale/libc/C", 0xbffff6c4) = -1 ENOENT (No such file or directory)
08:55:24 stat("/usr/share/locale/C/libc.cat", 0xbffff6c4) = -1 ENOENT (No such file or directory)
08:55:24 stat("/usr/local/share/locale/C/libc.cat", 0xbffff6c4) = -1 ENOENT (No such file or directory)
08:55:24 socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
08:55:24 sigaction(SIGUSR1, {0x804a6b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}, 0x42029098) = 0
08:55:24 socket(PF_INET, SOCK_RAW, IPPROTO_RAW) = 4
08:55:24 setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
08:55:24 getpid()                       = 5119
08:55:24 getpid()                       = 5119
08:55:24 shmget(5361, 240, IPC_CREAT|0) = 65538
08:55:24 semget(5543, 1, IPC_CREAT|0x180|0600) = 65538
```

```
08:55:24 shmat(65538, 0, 0)           = 0x40008000
08:55:24 write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52) = 52
08:55:24 time([1044712524])           = 1044712524
08:55:24 close(0)                     = 0
08:55:24 sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x42029098) = 0
08:55:24 sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x42029098) = 0
08:55:24 sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x42029098) = 0
08:55:24 fork()                       = 5364
08:55:24 close(4)                     = 0
08:55:24 close(3)      = 0
08:55:24 semop(65538, 0xbffffb3c, 2)              = 0
08:55:24 shmdt(0x40008000)                    = 0
08:55:24 semop(65538, 0xbffffb3c, 1) = 0
08:55:24 _exit(0)                     = ?
```

"trace-atd.5364" File

(output of strace on the unknown binary on a RedHat 7.3 Linux system, with no external interactions)

```
08:55:24 setsid() = 5364
08:55:24 open("/dev/tty", O_RDWR) = -1 ENXIO (No such device or address)
08:55:24 chdir("/tmp") = 0
08:55:24 umask(0) = 022
08:55:24 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}, 0x42029098) = 0
08:55:24 alarm(3600)                    = 0
08:55:24 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}, 0x42029098) = 0
08:55:24 read(3, 0x804c78c, 84)         = ? ERESTARTSYS (To be restarted)
09:55:24 --- SIGALRM (Alarm clock) ---
```

"trace-atd-42.32159" file

(output of strace on unknown binary on a RedHat 4.2 Linux system).

```
18:26:58 setsid()                    = 32159
18:26:58 open("/dev/tty", O_RDWR)       = -1 ENXIO (No such device or address)
18:26:58 chdir("/tmp")                  = 0
18:26:58 umask(0)                    = 022
18:26:58 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
18:26:58 alarm(3600)                 = 0
18:26:58 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
18:26:58 read(4, 0x804c78c, 84)         = ? ERESTARTSYS (To be restarted)
19:26:58 --- SIGALRM (Alarm clock) ---
19:26:58 alarm(0)                    = 0
19:26:58 time([1047256018])             = 1047256018
19:26:58 semop(0xb, 0x2, 0, 0xbffffbec) = 0
19:26:58 semop(0xb, 0x1, 0, 0xbffffbf4) = 0
19:26:58 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
19:26:58 alarm(3600)                 = 0
19:26:58 sigreturn()                 = ? (mask now [])
19:26:58 read(4, 0x804c78c, 84)         = ? ERESTARTSYS (To be restarted)
19:28:46 --- SIGTERM (Terminated) ---
19:28:46 +++ killed by SIGTERM +++
```

"trace-atd-42.26086" file

```
13:26:02 setsid() = 26086
13:26:02 open("/dev/tty", O_RDWR) = -1 ENXIO (No such device or address)
13:26:02 chdir("/tmp")                  = 0
13:26:02 umask(0)                       = 022
13:26:02 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
13:26:02 alarm(3600)                    = 0
13:26:02 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
13:26:02 read(4, "E\0\0<@\214\0\0@\1\273\350\300\250"..., 84) = 60
13:26:12 read(4, "E\0\0<@\214\0\0@\1\273\350\300\250"..., 84) = 60
13:26:13 read(4, "E\0\0<@\214\0\0@\1\273\350\300\250"..., 84) = 60
13:26:14 read(4, "E\0\0<@\214\0\0@\1\273\350\300\250"..., 84) = 60
13:26:15 read(4, 0x804c78c, 84)         = ? ERESTARTSYS (To be restarted)
```

"trace-atd-42.28418" file

(output of strace on the 1st child process of the unknown binary on a RedHat 4.2 Linux system, with LOKI2 client as the external interaction)

```
00:20:15 setsid()                       = 28418
00:20:15 open("/dev/tty", O_RDWR)       = -1 ENXIO (No such device or address)
00:20:15 chdir("/tmp")                  = 0
00:20:15 umask(0)                       = 022
00:20:15 sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
00:20:15 alarm(3600)                    = 0
00:20:15 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
00:20:15 read(4, "E\0\0T\261\355\0\0@\1\312\271\177"..., 84) = 84
00:20:21 fork()                         = 28422
00:20:22 read(4, "E\0\0T\261\356\0\0@\1\312\270\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\360\0\0@\1\312\266\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\361\0\0@\1\312\265\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\364\0\0@\1\312\262\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\365\0\0@\1\312\261\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\366\0\0@\1\312\260\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\372\0\0@\1\312\254\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\373\0\0@\1\312\253\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\374\0\0@\1\312\252\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\375\0\0@\1\312\251\177"..., 84) = 84
00:20:22 read(4, "E\0\0T\261\376\0\0@\1\312\250\177"..., 84) = 84
00:20:22 --- SIGCHLD (Child exited) ---
00:20:22 wait4(-1, [WIFEXITED(s) && WEXITSTATUS(s) == 0], 0, NULL) = 28422
00:20:22 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
00:20:22 sigreturn()                    = ? (mask now [])
00:20:22 read(4, 0x804c78c, 84)         = ? ERESTARTSYS (To be restarted)
00:20:42 --- SIGTERM (Terminated) ---
00:20:42 +++ killed by SIGTERM +++
```

"trace-atd-42.28422" file

(output of strace on the 2nd child process of the unknown binary on a RedHat 4.2 Linux system, with LOKI2 client as the external interaction)

```
00:20:22 semop(0x8, 0x2, 0, 0xbffffcfc) = 0
00:20:22 time(NULL)                 = 1047187222
00:20:22 semop(0x8, 0x1, 0, 0xbffffd00) = 0
00:20:22 pipe([0, 6])               = 0
00:20:22 fork()                     = 28423
00:20:22 close(6)                   = 0
00:20:22 fstat(0, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x400d3000
00:20:22 read(0, "install.log\nlibtermcap-2.0.8-4."..., 4096) = 89
00:20:22 --- SIGCHLD (Child exited) ---
00:20:22 wait4(-1, [WIFEXITED(s) && WEXITSTATUS(s) == 0], 0, NULL) = 28423
00:20:22 sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},
{SIG_DFL}) = 0
00:20:22 sigreturn()                = ? (mask now [])
00:20:22 oldselect(1, NULL, NULL, NULL, {0, 100}) = 0 (Timeout)
00:20:22 sendto(5, "E\0\0T\0\0\0\0@\1\0\0\0\0\0\177"..., 84, 0, {sin_family=AF_INET,
sin_port=htons(832), sin_addr=inet_addr("127.0.0.1")}, 16) = 84
00:20:22 oldselect(1, NULL, NULL, NULL, {0, 100})          = 0 (Timeout)
00:20:22 sendto(5, "E\0\0T\0\0\0\0@\1\0\0\0\0\0\177"..., 84, 0, {sin_family=AF_INET,
sin_port=htons(832), sin_addr=inet_addr("127.0.0.1")}, 16) = 84
00:20:22 oldselect(1, NULL, NULL, NULL, {0, 100})          = 0 (Timeout)
00:20:22 sendto(5, "E\0\0T\0\0\0\0@\1\0\0\0\0\0\177"..., 84, 0, {sin_family=AF_INET,
sin_port=htons(832), sin_addr=inet_addr("127.0.0.1")}, 16) = 84
00:20:22 oldselect(1, NULL, NULL, NULL, {0, 100})          = 0 (Timeout)
00:20:22 sendto(5, "E\0\0T\0\0\0\0@\1\0\0\0\0\0\177"..., 84, 0, {sin_family=AF_INET,
sin_port=htons(832), sin_addr=inet_addr("127.0.0.1")}, 16) = 84
00:20:22 read(0, "", 4096)          = 0
00:20:22 oldselect(1, NULL, NULL, NULL, {0, 100})          = 0 (Timeout)
00:20:22 sendto(5, "E\0\0T\0\0\0\0@\1\0\0\0\0\0\177"..., 84, 0, {sin_family=AF_INET,
sin_port=htons(0), sin_addr=inet_addr("127.0.0.1")}, 16) = 84
00:20:22 semop(0x8, 0x2, 0, 0xbffffcf8) = 0
00:20:22 time(NULL)                 = 1047187222
00:20:22 semop(0x8, 0x1, 0, 0xbffffcfc) = 0
00:20:22 semop(0x8, 0x2, 0, 0xbffffd00) = 0
00:20:22 time(NULL)                 = 1047187222
00:20:22 semop(0x8, 0x1, 0, 0xbffffd00) = 0
00:20:22 close(5)                   = 0
00:20:22 close(4)                   = 0
00:20:22 semop(0x8, 0x2, 0, 0xbffffce4) = 0
00:20:22 shmdt(0x40007000)          = 0
00:20:22 semop(0x8, 0x1, 0, 0xbffffce4) = 0
00:20:22 _exit(0)                   = ?
```

"trace-atd-42.28423" file

(Output of strace on the 3rd child process of the unknown binary on a RedHat 4.2 Linux system, with LOKI2 client as the external interaction)

```
00:20:22 close(0) = 0
00:20:22 dup2(6, 1) = 1
00:20:22 close(6) = 0
00:20:22 execve("/bin/sh", ["sh", "-c", "ls\n"], [/* 17 vars */]) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40006000
00:20:22 mprotect(0x8048000, 277863, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
00:20:22 stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=2050, ...}) = 0
00:20:22 open("/etc/ld.so.cache", O_RDONLY)                    = 0
00:20:22 mmap(0, 2050, PROT_READ, MAP_SHARED, 0, 0) = 0x40007000
00:20:22 close(0)                    = 0
00:20:22 open("/lib/libtermcap.so.2.0.8", O_RDONLY) = 0
00:20:22 read(0, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0"..., 4096) = 4096
00:20:22 mmap(0, 12288, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40008000
00:20:22 mmap(0x40008000, 7276, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 0, 0) =
0x40008000
00:20:22 mmap(0x4000a000, 3496, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 0,
0x1000) = 0x4000a000
00:20:22 close(0)                    = 0
00:20:22 mprotect(0x40008000, 7276, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
00:20:22 open("/lib/libc.so.5.3.12", O_RDONLY) = 0
00:20:22 read(0, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0"..., 4096) = 4096
00:20:22 mmap(0, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x4000b000
00:20:22 mmap(0x4000b000, 599154, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 0, 0)
= 0x4000b000
00:20:22 mmap(0x4009e000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 0,
0x92000) = 0x4009e000
00:20:22 mmap(0x400a4000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a4000
00:20:22 close(0)                    = 0
00:20:22 mprotect(0x4000b000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
00:20:22 munmap(0x40007000, 2050)      = 0
00:20:22 mprotect(0x8048000, 277863, PROT_READ|PROT_EXEC) = 0
00:20:22 mprotect(0x40008000, 7276, PROT_READ|PROT_EXEC) = 0
00:20:22 mprotect(0x4000b000, 599154, PROT_READ|PROT_EXEC) = 0
00:20:22 personality(PER_LINUX)        = 0
00:20:22 geteuid()                    = 0
00:20:22 getuid()                    = 0
00:20:22 getgid()                    = 0
00:20:22 getegid()                    = 0
00:20:22 getuid()                    = 0
00:20:22 getgid()                    = 0
00:20:22 geteuid()                    = 0
00:20:22 getegid()                    = 0
00:20:22 brk(0x8092938)                = 0x8092938
00:20:22 brk(0x8093000)                = 0x8093000
00:20:22 time(NULL)                    = 1047187222
00:20:22 sigaction(SIGCHLD, {SIG_DFL}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGCHLD, {SIG_DFL}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGINT, {SIG_DFL}, {SIG_DFL}) = 0
```

```
00:20:22 sigaction(SIGINT, {SIG_DFL}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGQUIT, {SIG_DFL}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGQUIT, {SIG_DFL}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGHUP, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGINT, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGILL, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGTRAP, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGABRT, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGFPE, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGBUS, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGSEGV, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGPIPE, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGALRM, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGTERM, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGXCPU, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGXFSZ, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGVTALRM, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV
USR2 PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGPROF, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGUSR1, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigaction(SIGUSR2, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1 SEGV USR2
PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}, {SIG_DFL}) = 0
00:20:22 sigprocmask(SIG_BLOCK, NULL, []) = 0
00:20:22 sigaction(SIGQUIT, {SIG_IGN}, {SIG_DFL}) = 0
00:20:22 open("/etc/nsswitch.conf", O_RDONLY) = 0
00:20:22 brk(0x8096000)                 = 0x8096000
00:20:22 fstat(0, {st_mode=S_IFREG|0644, st_size=1208, ...}) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40007000
00:20:22 read(0, "#\n# /etc/nsswitch.conf\n#\n# An"..., 4096) = 1208
00:20:22 read(0, "", 4096)              = 0
00:20:22 close(0)                       = 0
00:20:22 munmap(0x40007000, 4096)       = 0
00:20:22 open("/etc/passwd", O_RDONLY)  = 0
00:20:22 fstat(0, {st_mode=S_IFREG|0644, st_size=585, ...}) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40007000
00:20:22 read(0, "root:.n1kkKAgsCNfs:0:0:root:/roo"..., 4096) = 585
00:20:22 lseek(0, -541, SEEK_CUR)       = 44
00:20:22 close(0)                       = 0
00:20:22 munmap(0x40007000, 4096)       = 0
```

```
00:20:22 uname({sys="Linux", node="dhcppc0", ...}) = 0
00:20:22 brk(0x8097000)                 = 0x8097000
00:20:22 lstat(".", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=1024, ...}) = 0
00:20:22 lstat("/", {st_mode=S_IFDIR|0755, st_size=1024, ...}) = 0
00:20:22 lstat("..", {st_mode=S_IFDIR|0755, st_size=1024, ...}) = 0
00:20:22 stat("..", {st_mode=S_IFDIR|0755, st_size=1024, ...}) = 0
00:20:22 open("..", O_RDONLY)           = 0
00:20:22 fcntl(0, F_SETFD, FD_CLOEXEC)  = 0
00:20:22 brk(0x8099000)                 = 0x8099000
00:20:22 getdents(0, /* 16 entries */, 4096) = 260
00:20:22 lstat("../tmp", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=1024, ...}) = 0
00:20:22 close(0)                       = 0
00:20:22 getpid()                       = 28423
00:20:22 getppid()                      = 28422
00:20:22 stat(".", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=1024, ...}) = 0
00:20:22 stat("/bin/sh", {st_mode=S_IFREG|0755, st_size=300668, ...}) = 0
00:20:22 getpgrp()                      = 28418
00:20:22 fcntl(-1, F_SETFD, FD_CLOEXEC) = -1 EBADF (Bad file number)
00:20:22 sigaction(SIGCHLD, {0x805bc10, [], 0}, {SIG_DFL}) = 0
00:20:22 stat(".", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=1024, ...}) = 0
00:20:22 stat("/bin/ls", {st_mode=S_IFREG|0755, st_size=49432, ...}) = 0
00:20:22 sigaction(SIGHUP, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGILL, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGTRAP, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGABRT, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGFPE, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGBUS, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGSEGV, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGPIPE, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGALRM, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGTERM, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGXCPU, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGXFSZ, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGVTALRM, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGPROF, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGUSR1, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGUSR2, {SIG_DFL}, NULL) = 0
00:20:22 sigaction(SIGINT, {SIG_DFL}, {0x804b510, [HUP INT ILL TRAP ABRT BUS FPE USR1
SEGV USR2 PIPE ALRM TERM XCPU XFSZ VTALRM PROF], 0}) = 0
00:20:22 sigaction(SIGQUIT, {SIG_DFL}, {SIG_IGN}) = 0
00:20:22 sigaction(SIGCHLD, {SIG_DFL}, {0x805bc10, [], 0}) = 0
00:20:22 execve("/bin/ls", ["ls"], [/* 16 vars */]) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40006000
00:20:22 mprotect(0x8048000, 46847, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
00:20:22 stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=2050, ...}) = 0
00:20:22 open("/etc/ld.so.cache", O_RDONLY) = 0
00:20:22 mmap(0, 2050, PROT_READ, MAP_SHARED, 0, 0) = 0x40007000
00:20:22 close(0)                       = 0
00:20:22 open("/lib/libc.so.5.3.12", O_RDONLY) = 0
00:20:22 read(0, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"..., 4096) = 4096
00:20:22 mmap(0, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40008000
00:20:22 mmap(0x40008000, 599154, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 0, 0)
= 0x40008000
00:20:22 mmap(0x4009b000, 22664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 0,
0x92000) = 0x4009b000
```

```
00:20:22 mmap(0x400a1000, 200812, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a1000
00:20:22 close(0)                  = 0
00:20:22 mprotect(0x40008000, 599154, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
00:20:22 munmap(0x40007000, 2050)      = 0
00:20:22 mprotect(0x8048000, 46847, PROT_READ|PROT_EXEC) = 0
00:20:22 mprotect(0x40008000, 599154, PROT_READ|PROT_EXEC) = 0
00:20:22 personality(PER_LINUX)        = 0
00:20:22 geteuid()                 = 0
00:20:22 getuid()                  = 0
00:20:22 getgid()                  = 0
00:20:22 getegid()                 = 0
00:20:22 open(0xbffffc84, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 open(0xbffffc84, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 open(0xbffffc84, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 open(0xbffffc8c, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 open(0xbffffc84, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 open(0xbffffc88, O_RDONLY)    = -1 ENOENT (No such file or directory)
00:20:22 brk(0x8054bf4)              = 0x8054bf4
00:20:22 brk(0x8055000)              = 0x8055000
00:20:22 time(NULL)                = 1047187222
00:20:22 ioctl(1, TCGETS, 0xbffffd6c)  = -1 EINVAL (Invalid argument)
00:20:22 ioctl(1, TIOCGWINSZ, 0xbffffda4) = -1 EINVAL (Invalid argument)
00:20:22 brk(0x8058000)              = 0x8058000
00:20:22 stat(".", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=1024, ...}) = 0
00:20:22 open(".", O_RDONLY)           = 0
00:20:22 fcntl(0, F_SETFD, FD_CLOEXEC)  = 0
00:20:22 getdents(0, /* 6 entries */, 4096) = 164
00:20:22 getdents(0, /* 0 entries */, 4096) = 0
00:20:22 close(0)                  = 0
00:20:22 fstat(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
00:20:22 mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40007000
00:20:22 write(1, "install.log\nlibtermcap-2.0.8-4."..., 89) = 89
00:20:22 close(1)                  = 0
00:20:22 munmap(0x40007000, 4096)      = 0
00:20:22 _exit(0)                  = ?
```

Last Updated: November 15th, 2018

Upcoming SANS Forensics Training

SANS Paris November 2018	Paris, France	Nov 19, 2018 - Nov 24, 2018	Live Event
SANS November Singapore 2018	Singapore, Singapore	Nov 19, 2018 - Nov 24, 2018	Live Event
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS San Francisco Fall 2018	San Francisco, CA	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS Austin 2018	Austin, TX	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS Khobar 2018	Khobar, Kingdom Of Saudi Arabia	Dec 01, 2018 - Dec 06, 2018	Live Event
SANS Nashville 2018	Nashville, TN	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Frankfurt 2018	Frankfurt, Germany	Dec 10, 2018 - Dec 15, 2018	Live Event
SANS Cyber Defense Initiative 2018	Washington, DC	Dec 11, 2018 - Dec 18, 2018	Live Event
Cyber Defense Initiative 2018 - FOR500: Windows Forensic Analysis	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR585: Advanced Smartphone Forensics	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Mentor Session - FOR500	Phoenix, AZ	Jan 11, 2019 - Feb 15, 2019	Mentor
SANS Threat Hunting London 2019	London, United Kingdom	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Amsterdam January 2019	Amsterdam, Netherlands	Jan 14, 2019 - Jan 19, 2019	Live Event
Mentor Session - FOR508	Copenhagen, Denmark	Jan 16, 2019 - Mar 09, 2019	Mentor
Cyber Threat Intelligence Summit & Training 2019	Arlington, VA	Jan 21, 2019 - Jan 28, 2019	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201901,	Jan 21, 2019 - Feb 27, 2019	vLive
SANS Miami 2019	Miami, FL	Jan 21, 2019 - Jan 26, 2019	Live Event
Mentor Session - FOR585	Tampa, FL	Jan 24, 2019 - Mar 07, 2019	Mentor
SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
Security East 2019 - FOR585: Advanced Smartphone Forensics	New Orleans, LA	Feb 04, 2019 - Feb 09, 2019	vLive
SANS London February 2019	London, United Kingdom	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Anaheim 2019	Anaheim, CA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS vLive - FOR578: Cyber Threat Intelligence	FOR578 - 201902,	Feb 11, 2019 - Mar 20, 2019	vLive
Community SANS Madrid FOR610 (in Spanish)	Madrid, Spain	Feb 11, 2019 - Feb 16, 2019	Community SANS
SANS Northern VA Spring- Tysons 2019	Vienna, VA	Feb 11, 2019 - Feb 16, 2019	Live Event
SANS Dallas 2019	Dallas, TX	Feb 18, 2019 - Feb 23, 2019	Live Event
SANS New York Metro Winter 2019	Jersey City, NJ	Feb 18, 2019 - Feb 23, 2019	Live Event