Practical Assignment v1.5
Part One – Analyze An Unknown Image
Part Two, Option One – Perform Forensic Analysis On A System

YOU CAN RUN, BUT YOU CAN'T HIDE:
A WALKTHROUGH OF TWO COMPUTER FORENSICS CASES

PETER C. HEWITT

Submitted Thursday, November 4, 2004

Table of Contents

Appendix B: IPCop Network Log for Honeypot Compromise 53

Abstract and Statement of Purpose

The objectives of this paper are to fulfill the requirements of the GIAC Certified Forensic Analyst (GCFA) Practical Assignment version 1.5. The paper is divided into two parts, both of which involve the forensic analysis of previously unknown computer data images, using forensically sound techniques. The first part of the paper will involve the analysis of an unknown image provided by the SysAdmin, Audit, Network, and Security (SANS) Institute website. The second part of the paper will show the forensic analysis of a potentially compromised system in an unknown state. Both examinations will utilize the forensic investigation methodology as outlined by the GCFA course material[1]:

1. Verification
2. System Description
3. Evidence Collection
4. Timeline Creation and Analysis
5. Operating System – Specific Media Analysis
6. Data Recovery
7. String Search
8. Reporting

The analysis will make use of the tools taught in the GCFA class, but will not be limited to them.

Grammatical and Stylistic Conventions

Throughout this paper, several pronouns will be used to avoid confusion between the investigator and the (alleged) perpetrators of computer mischief. "I", "myself" and "my" will hereinafter refer to the author of this paper as well as the person performing the forensic analyses. "We" will be used in situations where I assume the reader can draw conclusions based on the information placed in front of him or her (i.e. "we can see by the following facts that this computer has been compromised.") "S/he", the "assailant", "attacker" or "intruder" will represent the perpetrator in Part two of the paper (SANS has supplied fictitious names for the players in the Part One analysis.)

To enhance the readability of this paper, the following stylistic conventions will be used:

- My analysis and comments will appear in 12-point Arial type.
- File and program names will appear in "quotes" the first time they are mentioned.
- `System commands will appear in Courier New.`
- **`Machine responses to these commands will appear in Bold Courier New.`**
- Reproduced screenshots will contain their own fonts that may vary from the above.
- External IP addresses will be obfuscated to protect their real-life assignments.

Part 1 – Analyze An Unknown Image

Verification

Overview of the Case

In the fictitious scenario for Part One of this paper, I have been asked to analyze a floppy disk image by the security administrator of Ballard Industries, David Keen. Ballard designs and produces fuel cell batteries, and has recently come to suspect that its proprietary information, including customer lists, were given to one of its major competitors, Rift Inc. The floppy disk was seized from Ballard's lead process control engineer, Robert John Leszczynski, Jr., as he left the office on 26 April 2004. My charge from Mr. Keen is to analyze the image obtained from the floppy disk, determine what is on the disk and how it might have been used by Mr. Leszczynski, and provide a report stating what information, if any, had been compromised and how to determine if Mr. Leszczynski had compromised other systems at Ballard.

System Description

The system I used for my analysis in this case is a Toshiba Satellite P25 laptop computer, with a 3.2 GHz processor and 512MB of memory. I had previously installed Red Hat Linux 9 along with the original manufacturer's installation of the Windows XP Home Edition operating system, and installed additional security and forensic tools on the Linux side, for use in my SANS GCFA classes. Additionally, I utilized the Helix Incident Response and Forensics Live CD[2], a self-booting disc based on the Knoppix operating system (in which in turn is based on the Linux operating system) that contains pre-configured security and forensic tools and has the added advantage of not using nor altering the host system's hard drive, as all operations are performed in the computer's Random Access Memory (RAM). Were one of the files I was examining somehow begin executing, any damage it would cause would be limited to the computer's RAM, which would disappear when I turned the power off. To further prevent the potential spread of malicious code, I did not connect this computer to any others while performing my analysis. My system was made physically secure by virtue of it being in my house, under lock and alarm.

Evidence Collection

Examination Details

I received the floppy disk and a related chain-of-custody (COC) form from Mr. Keen. The COC form describes in detail whom has had possession of and contact with a particular piece of evidence since that evidence was seized. COC provides a measure of assurance that evidence was not tampered with, since its possession is known at all times. The details of the COC form follow:

- Evidence Tag #: fl-260404-RJL1
- Evidence Item Physical Description: 3.5 inch TDK floppy disk

At this point we should note that, for the purposes of this paper, since I was not able to actually possess the floppy disk, I downloaded an "image" file of the data contained on the disk from the SANS website. To ensure the data did not change from the floppy disk to the image file, SANS created a cryptographic "fingerprint" of the data using the

Message Digest 5 (MD5)[3] "hashing" algorithm. This algorithm produces a 32-character fingerprint that is unique to this specific set of data; changing even one character of the data would result in an observably different fingerprint output, so it is infeasible with today's technology (although theoretically possible) to produce two different files that would create the same fingerprint. The MD5 fingerprint is widely accepted as valid in court cases. The COC form contained the following additional data:

- Image Filename: fl-260404-RJL1.img
- Cryptographic (MD5) Fingerprint: d7641eb4da871d980adbe4d371eda2ad
- Compressed Image Filename: fl-260404-RJL1.img.gz

The file downloaded from SANS was not entitled "fl-260404-RJL1.img" as shown above, nor entitled "fl-260404-RJL1.img.gz" but rather "V1_5.gz." I needed to ensure the file I downloaded was the same file that Mr. Keen (SANS) intended me to analyze, but first I wanted to gather some information about the compressed file. The .gz file extension told me that the file was a compressed archive created by the "gzip" file compression utility. I used my copy of gunzip (which decompresses .gz files) on the Helix CD to first list the contents of the .gz archive. I used two options, "-v" (for verbose output) and "-l" (to list the contents.)

```
[root (helix)]# gunzip -v -l v1_5.gz
method  crc      date   time  compressed  uncompressed  ratio  uncompressed_name
defla   948edf93 Oct 30 12:30 502408      1474560       65.9%  v1_5
```

The .gz archive contains only one compressed file inside itself, the "v1_5" file. I next ran the "File" utility on the .gz archive. One of the tests File performs is an attempt to determine the file type of a given file by comparing data within a known location in the file to a listing of "magic numbers" used by common file formats; if a number matches, File assumes the file to be in that format.

```
[root (helix)]# file v1_5.gz
v1_5.gz: gzip compressed data, was "fl-260404-RJL1.img", from Unix
```

From the File output we can confirm that this file is indeed a gzip compressed archive. Additionally, we can see that the file appears to have been named "fl-260404-RJL1.img" (which is the name used on the SANS website) prior to compression, but has been renamed to output v1_5 when uncompressed.

To uncompress the file into something I could examine, I ran the gunzip utility once again, using a slightly different set of options: "-v" (for verbose output) and "-d" (to decompress the file.)

```
[root (helix)]# gunzip -v -d v1_5.gz
v1_5.gz:   65.9% -- replaced with v1_5
```

This means that the gzip file was replaced by its contents in the same directory. Now that I had the actual image file to examine, I wanted to ensure that it had not changed

since I received it from Mr. Keen. I used the "md5sum" utility to determine the MD5 fingerprint of the file I received and compared it to the fingerprint shown above.

```
[root (helix)]# md5sum -v v1_5
d7641eb4da871d980adbe4d371eda2ad  v1_5
```

We can see that the MD5 fingerprint of v1_5 exactly matches that of fl-260404-RJL1.img on the COC form above. I now ran File again on the uncompressed file to see what additional facts I could gather.

```
[root (helix)]# file v1_5
v1_5: x86 boot sector, code offset 0x3c, OEM-ID " mkdosfs", root
entries 224, sectors 2872 (volumes <=32 MB) , sectors/FAT 9,
serial number 0x408bed14, label: "RJL        ", FAT (12 bit)
```

From the output of the File command we can surmise that the image is probably of a floppy disk made by what used to be known as an "IBM-compatible machine", or now known as "Wintel" (Microsoft Windows/ DOS operating system, Intel central processing unit.) The items above supporting this conclusion are: "x86 boot sector" (reflecting the x86 series of Intel processors), and "sectors/FAT 9" and "FAT (12 bit)" which reflect the File Allocation Table (FAT) system used by Wintel systems for floppy disks. However, we will not know the actual types of files contained in the image until we extract them.

My next step in the analysis was to run Autopsy (version 2.03)[4]. According to the program's website, "The Autopsy Forensic Browser is a graphical interface to the command line digital forensic analysis tools in The Sleuth Kit. Together, The Sleuth Kit and Autopsy provide many of the same features as commercial digital forensics tools for the analysis of Windows and UNIX file systems."[5] Forensic practitioners generally trust Autopsy not to modify the data under examination; however, I took extra steps to ensure that this remained the case, as we will see in the following sections. Autopsy is pre-configured in the Helix distribution, so I activated the program from the desktop, bringing up the following window:

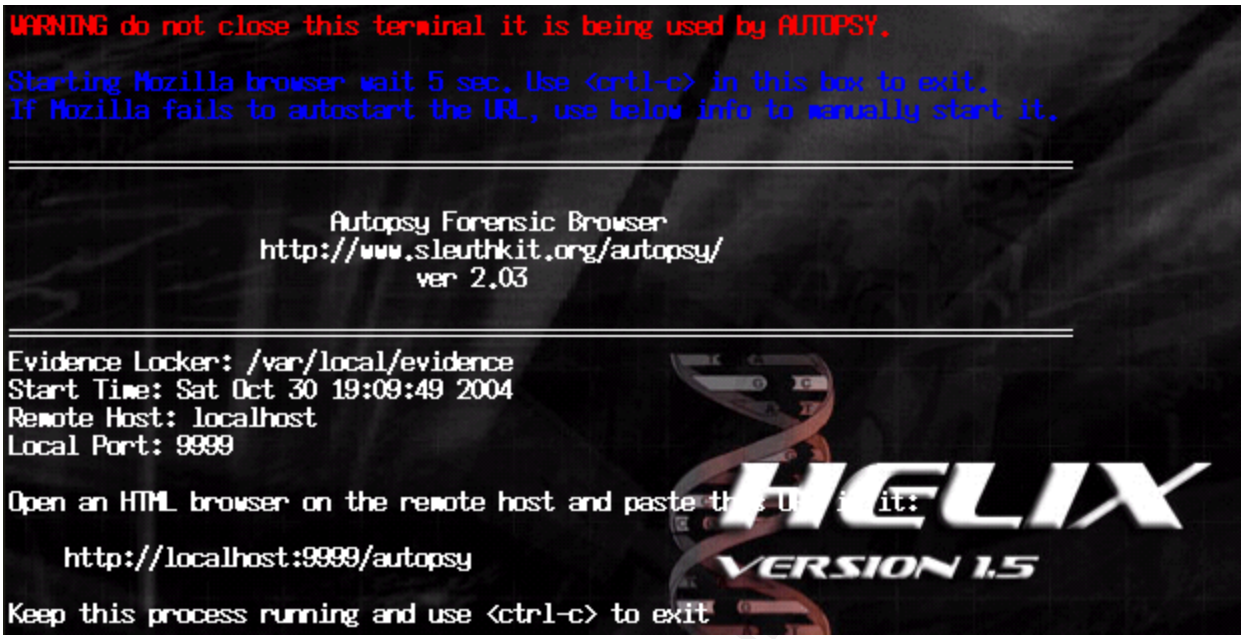


Figure 1: Autopsy startup window showing location of evidence locker

Note the location of the "evidence locker" (var/log/evidence): this is the folder location of all the logs, outputs and analyses that Autopsy generates. I made sure to copy this entire folder to a backup USB drive when I had finished my work, as all files contained in Helix would be erased from system memory once the computer had shut down.

To set up my analysis in Autopsy, I first created a new "case" in the system entitled "BallardFloppy" naming myself as the investigator. I then specified a "host" name; typically this is used when an entire computer system is being analyzed, as we will see in the second half of this paper, but for now I once again specified BallardFloppy as the host name. I then pointed Autopsy to the image (copied in from my USB drive) and had it create a "symbolic link" to the image file (to minimize the number of times the file would be copied over and possibly altered.) Autopsy, in "importing" (linking to) the image, once again created an MD5 fingerprint of the image, which I once again compared to the fingerprint specified by Mr. Keen. An exact match resulted, case notwithstanding:

Linking `/ramdisk/home/helix/v1_5` to `/var/local/evidence/BallardFloppy/BallardFloppy/images/v1_5`

Calculating MD5 of `images/v1_5` (this could take a while)
Current MD5: `D7641EB4DA871D980ADBE4D371EDA2AD`

Image: `/ramdisk/home/helix/v1_5` added to config file as `images/v1_5`

Figure 2: MD5 fingerprint calculation performed by Autopsy as it links to image file

We shall see in the following sections that Mr. Leszczynski was attempting to sell Ballard company secrets to another (unknown) entity. He does not appear to have been successful, as the floppy disk on which the information was contained was

seized before he could bring it outside of the company. However, a full investigation of the workstation that Mr. Leszczynski typically uses at Ballard should be made, as well as a review of any other removable media (floppy disks, CDs) in Mr. Leszczynski's work area. Additionally, steps (outlined in the following sections) should be taken to ascertain whether Mr. Leszczynski tampered with any other systems at Ballard, and if so, how it was done and what damage may have been caused.

Timeline Creation and Analysis

Image Details

Once I had the image file ready to go, I needed to decide what types of analyses I would have Autopsy perform. I first chose the main Autopsy file analysis by clicking "OK" at the Host Manager page, and was presented with the following options at the top of the screen:

Figure 3: Autopsy file analysis option tabs

I selected "File Analysis" and was presented with a listing of the files contained within the image. As described by Mr. Keen, the disk appeared to contain Microsoft Word documents of Ballard's various security policies: a little odd for a senior process control engineer to want to carry around, but not an outright indication of wrongdoing (save that Mr. Leszczynski was carrying them out of the company campus on a floppy disk.) I noted a total of 7 files contained within the image, and one volume label entry (the name given to the floppy disk by Mr. Leszczynski. Two of the files, both not Word documents like the others appeared to be, had been deleted, but were still recoverable.

Any file on a computer system has two components: the data stored in the file itself (a letter to Grandma, for example) and metadata, or "data about the data" (such as the type of file, the user who owns it, and the location of the file on the storage media.) Metadata has been compared by SANS instructor Rob Lee[6] to a card catalog in a library, where the computer files are the books and the metadata / card catalog shows their location on the shelf, publication date, etc. One of the most important pieces of metadata contained in a file is its timeline, or what's known as the MAC times. MAC stands for the last time a file was Modified (written), Accessed (read) or Changed (had its metadata written to.) These parameters differ by filesystem; in the FAT filesystem I believe was used on the floppy disk, the times are Written (modified), Accessed (same as before) and Created. These times can be extracted from an image by Autopsy and viewed, even for files that had been deleted. The figure below shows the files noted by Autopsy, their timeline parameters, size, and userID and group membership of the person that created.

Current Directory: C:\

ADD NOTE | GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID	META
✓	r / r	_ndex.htm	2004.04.23 10:53:56 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:47:36 (MDT)	727	0	0	28
	r / r	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)	2004.04.23 14:10:50 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:44 (MDT)	22528	0	0	27
✓	r / r	CamShell.dll (_AMSHELL.DLL)	2001.02.03 19:44:16 (MST)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:18 (MDT)	36864	0	0	5
	r / r	Information_Sensitivity_Policy.doc (INFORM~1.DOC)	2004.04.23 14:11:10 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:20 (MDT)	42496	0	0	9
	r / r	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)	2004.04.22 16:31:06 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:24 (MDT)	33423	0	0	17
	r / r	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)	2004.04.22 16:31:06 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:22 (MDT)	32256	0	0	13
	r / r	Password_Policy.doc (PASSWO~1.DOC)	2004.04.23 11:55:26 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:26 (MDT)	307935	0	0	20
	r / r	Remote_Access_Policy.doc (REMOTE~1.DOC)	2004.04.23 11:54:32 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:36 (MDT)	215895	0	0	23
	r / r	RJL (Volume Label Entry)	2004.04.25 10:53:40 (MDT)	2004.04.25 00:00:00 (MDT)	2004.04.25 10:53:40 (MDT)	0	0	0	3

Figure 4: Autopsy list of files contained in image

Autopsy highlights the deleted files in red and puts a checkmark in the first column (labeled "Del".) The "Type" column shows whether the entry is a directory or a file ("in" stands for inode, which is a type of metadata.) The next three columns show the timeline parameters. We find it interesting that the file creation times appear to be subsequent to the file written and accessed times. This is an artifact of Microsoft FAT file systems where when a file is copied from one system to another (say, from a hard drive to a floppy disk) the file creation time is updated while the other two parameters are retained. The file creation times above are within seconds of one another, meaning that either Mr. Leszczynski is a *very* fast typist or he copied these files from elsewhere.

The next two columns show no data ("0") for the userID and group membership of the files' creator. This too is par for the course for the Microsoft FAT filesystem, which does not keep track of user and group IDs relating to files (subsequent Microsoft file and operating systems do record this data, but only FAT is currently usable on floppy disks.)

The final column shows the location of the metadata related to each filename. By clicking on these we find information about each file, including the file type, MAC times, and the physical sectors of the disk on which the actual data resides.

In the tradition of the classic Sesame Street song "One of these is not like the others" it seems rather odd that the two deleted files are not of the same type as the other Word document files. The first deleted file, "_ndex.htm", appears to be a web page. The Index page is usually the first page a person comes across when s/he visits a website, providing a gateway to the rest of the site. The second deleted file, "CamShell.dll

(_AMSHELL.DLL) appears to be a Dynamic Link Library (DLL) commonly used by Microsoft Windows-based executable files. I made a mental note to pay special attention to these two files as my investigation progressed.

Seeing the timeline for each file is interesting, but a more complete picture is obtained when a full timeline is created. This function shows what a user did to each file in sequence, like a trail of footprints through a recently robbed bank. I returned to the Host page and selected "File Activity Timelines" from the main menu. Most people think that when they delete a file on their computer, it's gone for good, but that simply isn't the way systems work. A file can have one of three states:

- *Allocated*, where the data in the file is linked to an allocation entry in the file system, and can be read by a directory command or file browser;
- *Unallocated,* where the operating system has flagged the file as "deleted" but retains information on it, such as MAC times and physical location on the media, or
- *Unallocated Metadata Structures,* where the connection between the filename and its related metadata has been broken, but the connection between the metadata and the actual file data is preserved.

I requested that Autopsy look for file information in all three states. The results are displayed below:

```
Sat Feb 03 2001 19:44:16    36864 m.. -/-rwxrwxrwx 0        0        5        C:\/CamShell.dll (_AMSHELL.DLL) (deleted)
                            36864 m.. -rwxrwxrwx   0        0        5        <v1_5-_AMSHELL.DLL-dead-5>
Thu Apr 22 2004 16:31:06    33423 m.. -/-rwxrwxrwx 0        0        17       C:\/Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                            32256 m.. -/-rwxrwxrwx 0        0        13       C:\/Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Fri Apr 23 2004 10:53:56      727 m.. -/-rwxrwxrwx 0        0        28       C:\/_ndex.htm (deleted)
                              727 m.. -rwxrwxrwx   0        0        28       <v1_5-_ndex.htm-dead-28>
Fri Apr 23 2004 11:54:32   215895 m.. -/-rwxrwxrwx 0        0        23       C:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26   307935 m.. -/-rwxrwxrwx 0        0        20       C:\/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50    22528 m.. -/-rwxrwxrwx 0        0        27       C:\/Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10    42496 m.. -/-rwxrwxrwx 0        0        9        C:\/Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Sun Apr 25 2004 00:00:00        0 .a. -/-rwxrwxrwx 0        0        3        C:\/RJL         (Volume Label Entry)
Sun Apr 25 2004 10:53:40        0 m.c -/-rwxrwxrwx 0        0        3        C:\/RJL         (Volume Label Entry)
Mon Apr 26 2004 00:00:00    36864 .a. -rwxrwxrwx   0        0        5        <v1_5-_AMSHELL.DLL-dead-5>
                            33423 .a. -/-rwxrwxrwx 0        0        17       C:\/Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                           307935 .a. -/-rwxrwxrwx 0        0        20       C:\/Password_Policy.doc (PASSWO~1.DOC)
                            22528 .a. -/-rwxrwxrwx 0        0        27       C:\/Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
                              727 .a. -rwxrwxrwx   0        0        28       <v1_5-_ndex.htm-dead-28>
                            42496 .a. -/-rwxrwxrwx 0        0        9        C:\/Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
                            36864 .a. -/-rwxrwxrwx 0        0        5        C:\/CamShell.dll (_AMSHELL.DLL) (deleted)
                           215895 .a. -/-rwxrwxrwx 0        0        23       C:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
                            32256 .a. -/-rwxrwxrwx 0        0        13       C:\/Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
                              727 .a. -/-rwxrwxrwx 0        0        28       C:\/_ndex.htm (deleted)
Mon Apr 26 2004 09:46:18    36864 ..c -/-rwxrwxrwx 0        0        5        C:\/CamShell.dll (_AMSHELL.DLL) (deleted)
                            36864 ..c -rwxrwxrwx   0        0        5        <v1_5-_AMSHELL.DLL-dead-5>
Mon Apr 26 2004 09:46:20    42496 ..c -/-rwxrwxrwx 0        0        9        C:\/Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Mon Apr 26 2004 09:46:22    32256 ..c -/-rwxrwxrwx 0        0        13       C:\/Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Mon Apr 26 2004 09:46:24    33423 ..c -/-rwxrwxrwx 0        0        17       C:\/Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
Mon Apr 26 2004 09:46:26   307935 ..c -/-rwxrwxrwx 0        0        20       C:\/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36   215895 ..c -/-rwxrwxrwx 0        0        23       C:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44    22528 ..c -/-rwxrwxrwx 0        0        27       C:\/Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36      727 ..c -rwxrwxrwx   0        0        28       <v1_5-_ndex.htm-dead-28>
                              727 ..c -/-rwxrwxrwx 0        0        28       C:\/_ndex.htm (deleted)
```

From this timeline, it appears that whoever was in charge of the document files modified them on Friday, April 23, 2004, brought them all up into memory at the same time on Monday, April 26, and copied them over to another system nine hours thereafter. While this appears to be rather odd timing at first, we must keep in mind that we do not know the time zone in which the actions were committed, and thus Autopsy (using the current time zone in which it was running, Pacific Standard Time) may skew the results.

String Search

Forensic Details

Having not found much that I didn't already suspect out from the timeline analysis, I next turned to the keyword search capability available in Autopsy. I first returned to the File Analysis page, clicked on each file, and selected "Strings – Report". Strings, in computer parlance, are words or phrases. The Strings function in Autopsy looks for words in a given file of a certain minimum length (default is four) and lists them for the analyst's review. From these listings, an analyst can assemble a "dirty word list" of words that might prove useful in examining other files in the investigation. An example of this would be the word "MP3" when searching for evidence of illegally copied music.

A complete listing of the Strings output for each deleted file in the image is included at the end of this paper. I once again turned my attention to these deleted files. The first file, "_ndex.htm", appears to contain a reference to a Shockwave Flash file pertaining to Ballard Inc., as the Autopsy report content shows:

CONTENT

```
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;  charset=ISO-
8859-1">
<TITLE>Ballard</TITLE>
</HEAD>
<BODY bgcolor="#EDEDED">
<center>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"

codebase="http://download.macromedia.com/pub/shockwave/cabs/flash
/swflash.cab#version=6,0,0,0"
 WIDTH="800" HEIGHT="600" id="ballard" ALIGN="">
 <PARAM NAME=movie VALUE="ballard.swf"> <PARAM NAME=quality
VALUE=high> <PARAM NAME=bgcolor VALUE=#CCCCCC> <EMBED
src="ballard.swf" quality=high bgcolor=#CCCCCC  WIDTH="800"
HEIGHT="600" NAME="ballard" ALIGN=""
 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED
```

```
>
</OBJECT>
</center>
</BODY>
</HTML>
```

The code used in a web page, HyperText Markup Language (HTML), makes use of "framing" where pieces of human-readable information are surrounded by special code "tags" telling the computer how to display this information. In the output above, we can see that this is a complete web page, bookended as it is by the <HTML> and </HTML> tags. What would Mr. Leszczynski be doing with a Shockwave Flash file on his diskette? I added the values "swf" and "ballard.swf" to my dirty word list and moved on to the second deleted file, "_amShell.dll". As stated before, this appears to be a library file accessed by a Windows-based executable file, but the file itself does not appear to be on the floppy. I once again generated the Autopsy Strings report and examined it for interesting words:

CONTENT

```
(Same HTML code as in "_ndex.htm")
ll\SheCamouflageShell
ShellExt
VB5!
CamShell
BitmapShellMenu
CamouflageShell
CamouflageShell
Shell_Declares
Shell_Functions
ShellExt
modShellRegistry
kernel32
lstrcpyA
lstrlenA
ole32.dll
CLSIDFromProgID
StringFromGUID2
ReleaseStgMedium
shell32.dll
(multiple lines of code deleted for brevity)
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb
(remaining lines of code deleted for brevity)
```

Words that appear to repeat several times include Camouflage and "Shell." The most significant piece of data is the phrase "C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb" which suggests that Camouflage is was

originally written in Microsoft Visual Basic, a development platform used to create many Windows-based applications. At this point I decided to step away from the Autopsy application and access what is commonly believed to be a forensic analyst's most valuable resource, the search site www.Google.com. I began my Google search by putting in the "CamouflageShell" keyword, but the only results I retrieved were for physical camouflage products, including camper shells for pickup trucks. I next tried the actual filename I was analyzing, camshell.dll. That brought up a single web page, but one of extreme significance – it appeared to be a discussion board of people who appeared to believe that a program called Camouflage contained malicious code:

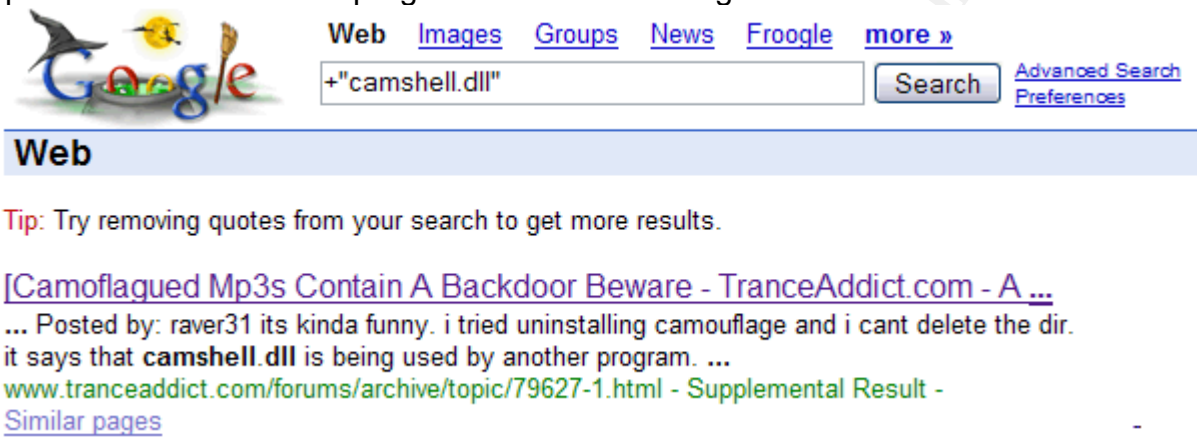


Figure 5: Google response for search on camshell.dll

When I went to referenced web page, I found a long list of people commenting on both the Camouflage program and the possibility that it contained malicious code. From the final post in this series, it turned out the user's problem was unrelated to Camouflage, but the discussion also gave me a tantalizing piece of information on the Camouflage program:

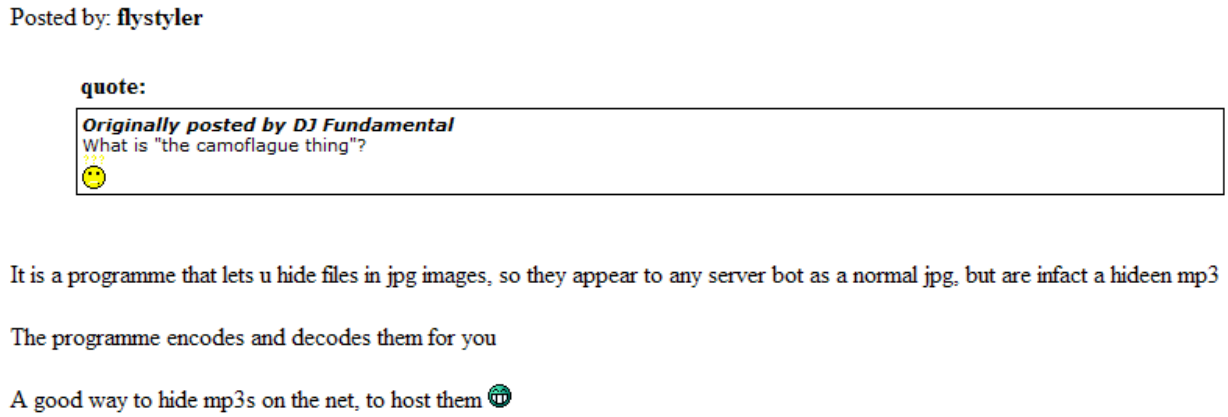


Figure 6: Discussion group explanation of Camouflage

This was very interesting – a program that could hide files inside other files. This concept is known as steganography, "The art and science of hiding information by embedding messages within other, seemingly harmless messages."[7] (webopedia reference) I added the word steganography to my search for Camouflage in Google,

and scored a direct hit:

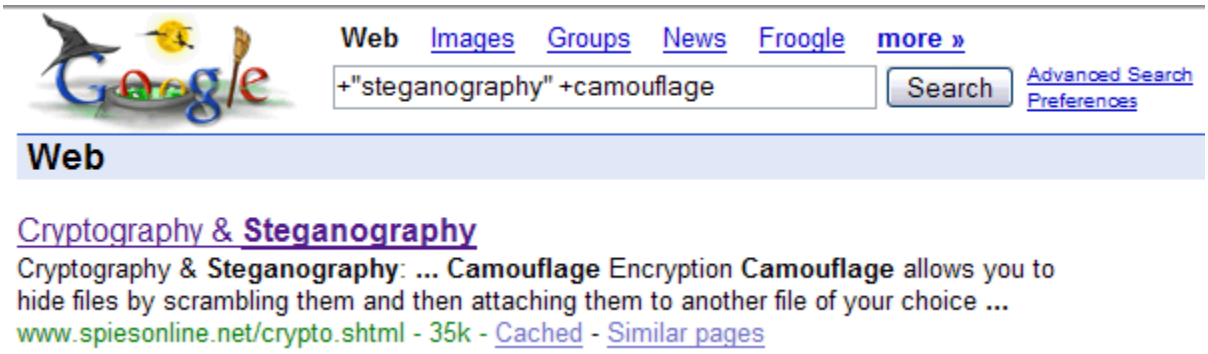


Figure 7: Google response for search on steganography and camouflage

I clicked over to the spiesonline.net website, which contained a link under "Camouflage Encryption" to another page at camouflage.unfiction.com[8]:

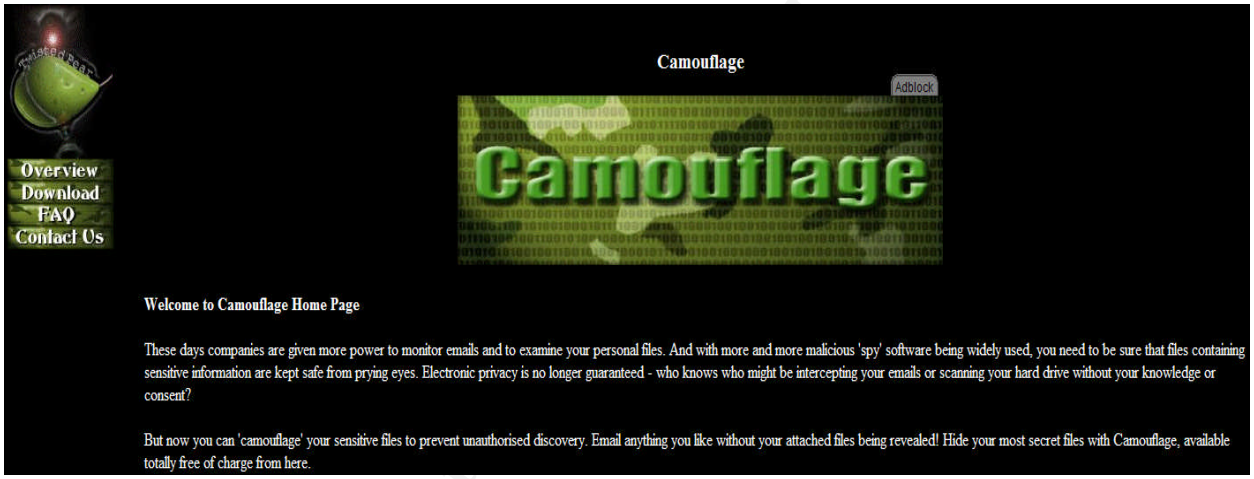


Figure 8: Camouflage main page at camouflage.unfiction.com

This appeared to be the main site for the program, and included a download section where I could retrieve a copy for my own testing. However, was this the same program that Mr. Leszczynski appeared to have used? If so, did he use it to hide data inside the files on the floppy, and if so, which ones? I continued to navigate around the unfiction website, and found a good explanation of the program in "Overview":

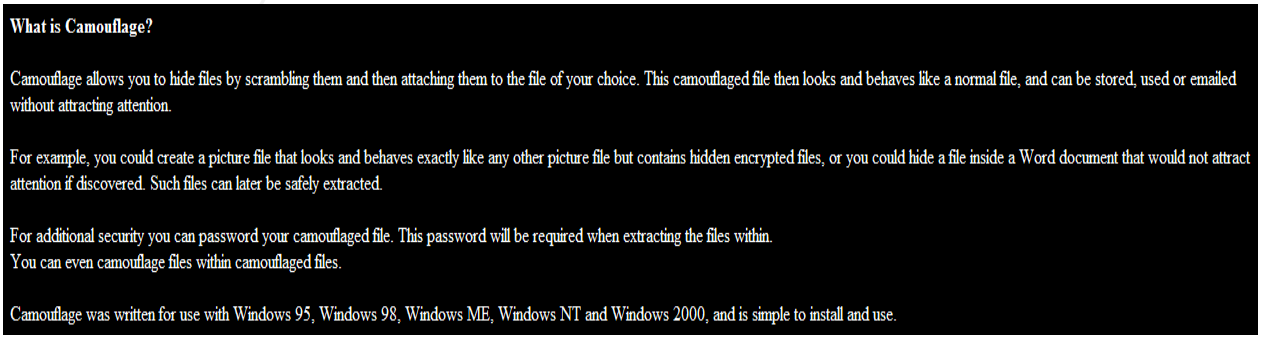


Figure 9: Camouflage overview page at camouflage.unfiction.com

Camouflage apparently had an option password-protection capability, which could present a problem as I obviously did not know any passwords to open Mr. Leszczynski's files. I also found from the "Download" that the program was no longer being developed by the authors:

Camouflage is free, but make sure you have read and understood the licence agreement before downloading it.

We regret that Camouflage is no longer supported or developed. The decision to abandon the project came after much debate, but we can no longer spare the resources needed to continue to provide quality service and support.

We hope that you will continue to use 'Camouflage', and although we will ensure that this site remains on-line for as long as possible, we encourage you to distribute Camouflage from your own 'unofficial' web site. Feel free to mirror these pages and to include the FAQ.

Please be aware that Unfiction.com is in no way affiliated with Camouflage software or it's developers.

The owners of the site have mirrored our site on their server, but any emails sent to them regarding Camouflage software will be deleted without being read.

Camouflage v1.2.1 - final version (Screenshot)
(Windows 95, 98, NT, 2000, ME)

Figure 10: Camouflage download page at camouflage.unfiction.com

The lack of support (along with a promise by the site owners to ignore any requests for information on the program) would certainly make it more difficult to determine if Mr. Leszczynski's suspected program and Camouflage were one in the same – one of the techniques commonly used by forensic specialists is to download a separate copy of a program's "source code" (the human-readable instructions that make up the program), "compile" it (convert the source code into something the computer can directly read, called "object code") and then compare the two programs running side-by-side. My problem here was twofold: a lack of the source code for Camouflage and the fact that the Camouflage program did not appear to exist on Mr. Leszczynski's disk, except in the form of camshell.dll. I nonetheless downloaded the "final version" (1.2.1) of Camouflage for review.

If Mr. Leszczynski did in fact use Camouflage to hide some as-yet-unknown information, how could I prove that he did so? The best way would be to look at the computer Mr. Leszczynski had performed his daily duties on and see if I could find the Camouflage program itself. Lacking that capability, I looked at the last time the related "_camshell.dll" file had been accessed from the timeline created by Autopsy. It matched the time of all the other files on the disk – midnight on Monday, April 26, 2004. This supports the theory that Mr. Leszczynski utilized the Camouflage program (and the .dll library that allowed to function) at the same time as he accessed the other files on the computer.

Operating System-Specific Media Analysis

Program Identification

Unfortunately, due to the lack of support by Camouflage's original authors, I was unable to locate the source code of the program for download and comparison to the .dll file I had in hand. I took several alternate steps to ensure that the executable program I had downloaded was the same as the one used by Mr. Leszczynski,

including performing a monitored test installation, and comparing the library file of the executed program with the same one located on Mr. Leszczynski's floppy.

Up until this point, the chances of a program I was analyzing somehow executing and damaging my system was remote, as I was using the Helix CD (which protects the hard drive of the host computer from being written to) and the programs appeared to run in a Microsoft Windows environment and thus would not execute in something Linux-based. However, I now *wanted* to execute the program, so I found an older server box (generic brand, AMD K2/350 processor, 256MB of RAM), installed a copy of Windows 2000 Advanced Server on it, and copied the downloaded Camouflage file onto it.

While I wanted to run the Camouflage program primarily to see if I could get any information out of Mr. Leszczynski's files, I also wanted to observe the program as it ran to see if it did anything else beside hide files inside of other files. Therefore, I used the following utilities on the Windows box:

- "Regmon"[9] – This program monitors changes to the Windows Registry (location in the Windows operating system where many configuration parameters are stored, in the form of "keys") to determine what changes the Camouflage installation or run program might make to the registry;
- "Winalysis"[10] – This program monitors changes to the Windows Registry similar to regmon, but instead of actively recording the changes like regmon does, "Winalysis" takes a snapshot of the registry before and after the execution of the program being scrutinized. Additionally, Winalysis looks at changes to files, user accounts, groups, services, and more.
- "Netstat" – A networking program installed with Windows 2000 that shows detail of all network connections on a computer and if a computer is listening for new connections. Similar to Winalysis I ran Netstat once before and after I ran Camouflage.

From the report output of these three programs I learned the following: although the installation and execution of Camouflage made some registry changes, these were consistent with the installation of a Windows executable that made use of context (right-click) menus as Camouflage does. Additionally, no changes were made to users, groups, services, or file sharing settings, nor were any network changes discovered by comparing the two Netstat outputs.

To corroborate my findings here, I once again investigated Google to see if anyone had done an analysis of the changes made by the Camouflage program, and found one at the SANS Institute by John Bartlett[11], a candidate for a different GIAC certificate. John noted, as I did, that the changes made to registry by Camouflage were minor. John took a further step of *un*installing the Camouflage program and noted that it left behind a number of changes it had made, including a list of all files on the computer that had been "camouflaged." This would prove invaluable to checking all the systems at Ballard (including Mr. Leszczynski's) for tampering. A simple check of the each

computer's registry for the following keys:

- HKEY_CURRENT_USER\Software\Camouflage\frmMain\CamouflageFileList
- HKEY_CURRENT_USER\Software\Camouflage\frmMain\uncamouflageFileList

would reveal what, if any, files had been "camouflaged" or "uncamouflaged."

Data Recovery

I also learned something disturbing while trying to figure out what was inside the files on the floppy; one or more of the files appeared to be protected with a password in Camouflage.

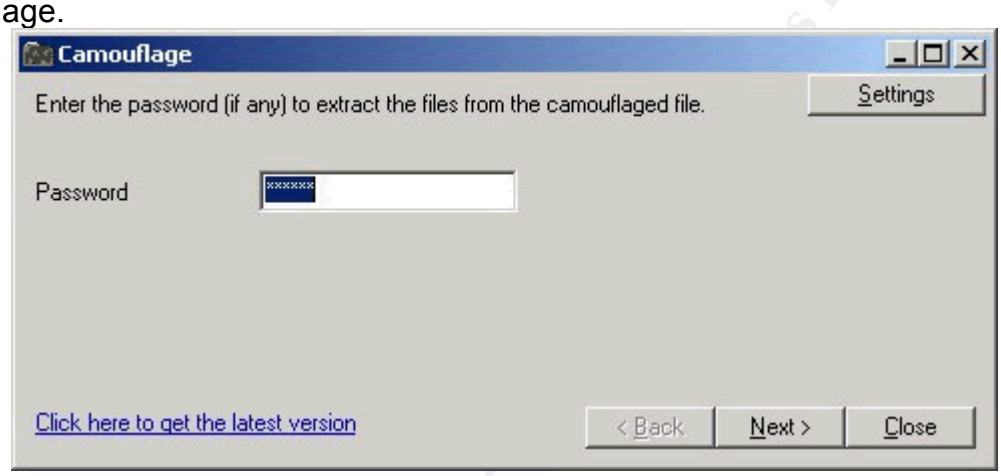


Figure 11 – Camouflage Password Challenge Screen

This meant that I couldn't open them and see what was inside unless I was able to guess the password. Several tries using both blank and obvious passwords ("Leszczynski", "ballard", "fuel cell" etc.) came up with nothing except the error window shown above. Camouflage didn't even do me the courtesy of telling me that my password was incorrect; I could have specified the wrong password, or the file could not have been modified by Camouflage. I returned to my Google search page and noted that the second item down seemed more promising:

Figure 12: Second Google response for search on steganography and camouflage

A website run by "Guillermito"[12] contains an overview of steganography in general, and specific information on the various steganographic programs available. One of these pages pays specific attention to Camouflage so I read with interest what he had to say.

Camouflage apparently utilizes a rather weak form of cryptography to hide its passwords. Cryptography is the science of scrambling information so it is only readable by those who have the correct key. Guillermito did a character-by-character analysis of several "camouflaged" files, viewing the data using a viewer based on the hexadecimal numbering system (known as a "hex editor.") Since files of the same type tend to have the same structure (components start and end at the same place with the same hex values) Guillermito reviewed several .JPG picture files that had been given (a) No password, (b) a password of four characters, and (c) a password consisting of the letter "a" repeated 255 times. He found the precise point in each file where the password was stored in an encrypted format. He also guessed (correctly) that the password had been stored and encrypted by using a mathematical operation known as exclusive or, or XOR. XOR compares two numbers, in this case the password and a never-changing set of numbers specified by Camouflage's authors. If the numbers are the same, XOR records a zero; if the numbers are different, XOR records a 1[13]. By using the 255-character password and the XOR function, Guillermito was able to determine the never-changing set of numbers, and thus could derive any password from its corresponding encrypted form.

Having read about this simple method of deriving passwords, I was heartened to see that Guillermito did me the favor of programming a simple application ("Camouflage Password Finder"[14]) that would do the calculation for me automatically. I downloaded

the program and returned to the file Analysis section of Autopsy. Now that I had a program that could determine whether the files on the floppy were "camouflaged" I had to extract the files from the floppy image. Fortunately, Autopsy eases this process by use of the "Export File" function. I was quickly able to bring out each file (even the deleted ones) into individual files that Windows could read.

I installed Guillermito's program on my Windows XP computer, reasoning that I would do the actual extraction of hidden files (if there were any) on the old Windows 2000 box, so if an extracted file tried to do damage I could simply erase the hard drive and reinstall without losing anything important. I tried the first file that had been extracted, the _ndex.htm file, with disappointing results:

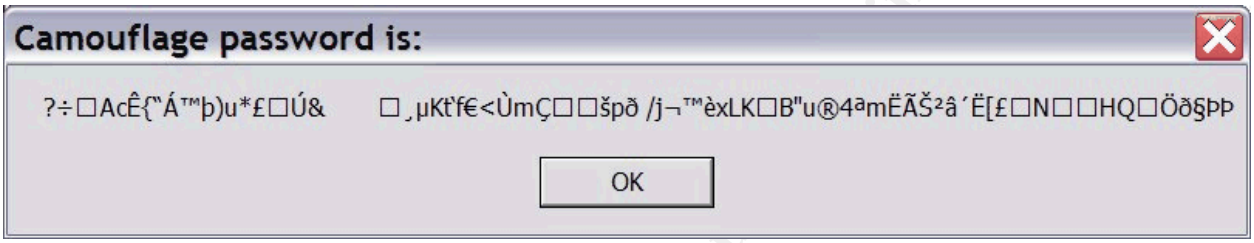


Figure 13: Camouflage Password Finder output for _ndex.htm

It was likely that this file did not contain a password, or was not even "camouflaged." Undaunted, I went back to the file listings and looked at the file sizes. I noted that only two of the files were over 100 kilobytes in size, and therefore the two of these would be the most likely location for hidden data, unless Camouflage included some sort of file compression, and I knew from John Bartlett's writing that this was not the case[15]. I instructed "Camouflage Password Finder" to look in the "Password_Policy" document, and this time was rewarded with a positive result:

Figure 14: Camouflage Password Finder output for Password_Policy.doc

I was also able to extract the password "Remote" for the file Remote_Access_Policy.doc. None of the other files appeared to have passwords. I moved over to the Windows 2000 system, right-clicked on the Remote_Access_Policy document, chose "Uncamouflage" from the resulting menu, entered the password I retrieved above, and had to restrain my shout of joy lest it wake up the neighbors:

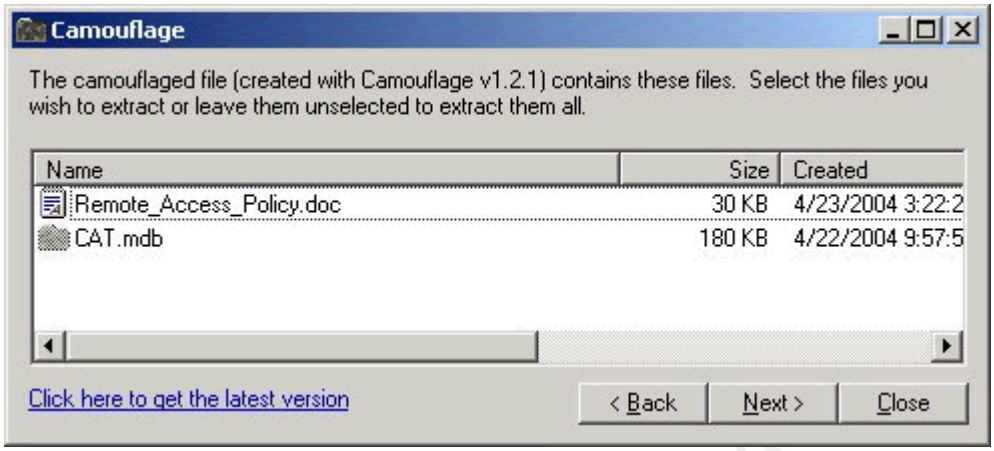


Figure 15 – Camouflage file extraction screen

Mr. Leszczynski seems to have been busy, to say the least – I found not only this .mdb (Microsoft Access Database) file, but also three graphic files from the Password_Policy document – one a reproduction of an article on fuel cells from the March 16, 2000 issue of the magazine *Nature*, and two diagrams of fuel cell Proton Exchange Membrane. The .mdb file is apparently a listing of some of Ballard's clients, and userIDs and passwords for them, possibly to get into a Ballard system:

Microsoft Access - [Clients : Table]

File Edit View Insert Format Records Tools Window Help

First	Last	Phone	Company	Address	Address1	City	State	Zipcode	Account	Password
Bob	Esposito	703-233-2048	Cook Labs	245 Main St		Alexandria	VA	20231	espomain	y4NSHMNf
Jerry	Jackson	410-677-7223	Double J's	11561 W. 27 St.		Baltimore	MD	20278	jack27st	JLbW3Pq5
David	Lee	866-554-0922	Tech Vision	300 Lone Grove Lane		Wichita	KS	30189	leetechv	O1A26a3k
Marie	Horton	800-234-king	King Labs, Inc.	700 King Labs Ave	Suite 900	Biloxi	MS	39533	hortking	Yk7Sr4pA
Lenny	Jones	877-Get-done	Quick Printing	99 E. Grand View Dr		Omaha	NE	56098	joneeast	868y48RH
Jeff	Hayes	404-893-5521	Big Sky First	90 Old Saw Mill Rd		Billings	MT	59332	hayeolds	3R30bb7i
Roger	Forrester	210-586-2312	TCFL	188 Greenville Rd		Austin	TX	77239	forrgree	si4OW8UV
Edward	Cash	212-562-0997	E & C Inc.	76 S. King St	Suite 300	Santa Barbara	CA	80124	cashking	OfBuQ1fC
Steve	Bei	616-833-0129	Island Labs	65 Kiwi Way		Honolulu	HA	93991	beikiwiw	JDH20u26
Jodie	Kelly		Data Movers	7256 Beerwah Ave.	Suite 110	Wetherby	U.K.	LS22 6RG	kellbeer	tmu0ENOk
Patrick	Roy		The Magic Lamp	4150 Regents Park	Row #170	Calgary	CAN	R4316DF	roythema	rJag6Q0O

Figure 16 – Microsoft Access database apparently of Ballard client names

The most damning piece of evidence against Mr. Leszczynski, however, came when I ran the "Camouflage Password Finder" against the extracted "Internal_Lab_Security_Policy.doc" file. It showed no password at all, and sure enough, when I simply hit "Enter" at the Camouflage password request window, it opened up a file called "Opportunity.txt", the contents of which are reproduced below:

```
I am willing to provide you with more information for a price.
I have included a sample of our Client Authorized Table database.
```

I have also provided you with our latest schematics not yet available. They are available as we discussed - "First Name". My price is 5 million.

Robert J. Leszczynski

We can see from this message that Mr. Leszczynski's intentions are clear, as well as his asking price.

To further match the downloaded Camouflage program with the remnant of the one on Mr. Leszczynski's floppy, I utilized a hex-editing program called "WinHex."[16] Since the _amshell.dll file appear to have been overwritten by the also-deleted _ndex.htm file on the floppy, the files would not match unless I could locate a common starting point between them. I was able to do so, and thus on both files I deleted all the data previous to these common points. I then re-saved the files under separate names (Camshell.dll and Camshell2.dll) and compared them to one another, as shown below:

CAMSHELL.DLL

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	2C	A4	0F	66	C0	5B	0F	66	6E	88	10	66	9E	00	10	66	¸¤.fÀ[.fn▌.f▌..f
00000010	C5	63	0D	66	0A	4A	02	66	49	54	02	66	F7	E0	0D	66	Åc.f.J.fIT.f÷à.f
00000020	73	00	10	66	81	54	0F	66	A3	A6	0E	66	08	5B	0F	66	s..f▌T.f£¦.f.[.f
00000030	B6	47	02	66	98	F7	0D	66	63	54	02	66	4E	AC	0E	66	¶G.f▌÷.fcT.fN¬.f
00000040	89	07	0E	66	BB	64	0D	66	1D	4C	02	66	35	54	0F	66	▌..f»d.f.L.f5T.f
00000050	90	11	0E	66	02	5A	0D	66	1F	43	02	66	CD	54	0F	66	▌..f.Z.f.C.fÍT.f
00000060	ED	6F	0F	66	D4	45	02	66	CD	55	0F	66	7C	87	02	66	ío.fÔE.fÍU.f|▌.f
00000070	D6	71	0F	66	94	A5	0F	66	1B	49	02	66	1B	A6	0E	66	Öq.f▌¥.f.I.f.¦.f
00000080	EB	44	02	66	77	D3	01	66	A2	13	0E	66	4F	48	02	66	ëD.fwÓ.f¢..fOH.f
00000090	CE	AB	10	66	1A	4A	02	66	78	B2	01	66	BD	5B	0F	66	Î«.f.J.fx².f½[.f
000000A0	07	CB	0E	66	0D	A1	10	66	9C	12	0E	66	0F	EA	02	66	.Ë.f.¡.f▌..f.ê.f
000000B0	C9	6D	0D	66	39	A6	0F	66	1D	59	0D	66	A0	7B	10	66	Ém.f9¦.f.Y.f {.f
000000C0	97	4E	02	66	9E	60	0D	66	38	B3	0E	66	50	58	0F	66	▌N.f▌`.f8³.fPX.f
000000D0	81	55	0F	66	5F	B4	0E	66	67	8D	10	66	20	C5	0E	66	▌U.f_´.fg▌.f Å.f
000000E0	DF	15	0E	66	8F	5B	0E	66	5C	AD	0E	66	CB	4D	0E	66	ß..f▌[.f\-.fËM.f
000000F0	7C	54	0E	66	4B	2D	0E	66	D1	A4	0F	66	26	83	0D	66	|T.fK-.fÑ¤.f&▌.f
00000100	4C	7C	10	66	AC	C2	01	66	01	55	0F	66	01	56	0F	66	L|.f¬Â.f.U.f.V.f
00000110	32	45	02	66	F0	59	0D	66	EC	48	02	66	35	55	0F	66	2E.fðY.fìH.f5U.f

CamShell2.dll

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	2C	A4	0F	66	C0	5B	0F	66	6E	88	10	66	9E	00	10	66	¸¤.fÀ[.fn▌.f▌..f
00000010	C5	63	0D	66	0A	4A	02	66	49	54	02	66	F7	E0	0D	66	Åc.f.J.fIT.f÷à.f
00000020	73	00	10	66	81	54	0F	66	A3	A6	0E	66	08	5B	0F	66	s..f▌T.f£¦.f.[.f
00000030	B6	47	02	66	98	F7	0D	66	63	54	02	66	4E	AC	0E	66	¶G.f▌÷.fcT.fN¬.f
00000040	89	07	0E	66	BB	64	0D	66	1D	4C	02	66	35	54	0F	66	▌..f»d.f.L.f5T.f
00000050	90	11	0E	66	02	5A	0D	66	1F	43	02	66	CD	54	0F	66	▌..f.Z.f.C.fÍT.f
00000060	ED	6F	0F	66	D4	45	02	66	CD	55	0F	66	7C	87	02	66	ío.fÔE.fÍU.f|▌.f
00000070	D6	71	0F	66	94	A5	0F	66	1B	49	02	66	1B	A6	0E	66	Öq.f▌¥.f.I.f.¦.f
00000080	EB	44	02	66	77	D3	01	66	A2	13	0E	66	4F	48	02	66	ëD.fwÓ.f¢..fOH.f
00000090	CE	AB	10	66	1A	4A	02	66	78	B2	01	66	BD	5B	0F	66	Î«.f.J.fx².f½[.f
000000A0	07	CB	0E	66	0D	A1	10	66	9C	12	0E	66	0F	EA	02	66	.Ë.f.¡.f▌..f.ê.f
000000B0	C9	6D	0D	66	39	A6	0F	66	1D	59	0D	66	A0	7B	10	66	Ém.f9¦.f.Y.f {.f
000000C0	97	4E	02	66	9E	60	0D	66	38	B3	0E	66	50	58	0F	66	▌N.f▌`.f8³.fPX.f
000000D0	81	55	0F	66	5F	B4	0E	66	67	8D	10	66	20	C5	0E	66	▌U.f_´.fg▌.f Å.f
000000E0	DF	15	0E	66	8F	5B	0E	66	5C	AD	0E	66	CB	4D	0E	66	ß..f▌[.f\-.fËM.f
000000F0	7C	54	0E	66	4B	2D	0E	66	D1	A4	0F	66	26	83	0D	66	|T.fK-.fÑ¤.f&▌.f
00000100	4C	7C	10	66	AC	C2	01	66	01	55	0F	66	01	56	0F	66	L|.f¬Â.f.U.f.V.f
00000110	32	45	02	66	F0	59	0D	66	EC	48	02	66	35	55	0F	66	2E.fðY.fìH.f5U.f

Page 1 of 114 Offset: 0

Figure 17 – Edited Camshell.dll files within WinHex editor

The files appeared to match exactly, but I needed to prove this. Fortunately, "WinHex" also contains an option that will calculate both MD5 fingerprints and another, more sophisticated fingerprint, Secure Hash Algorithm 1 (SHA-1.)[17] I therefore calculated the MD5 and SHA-1 fingerprints for both files and compared them, noting that they were

now exactly the same:

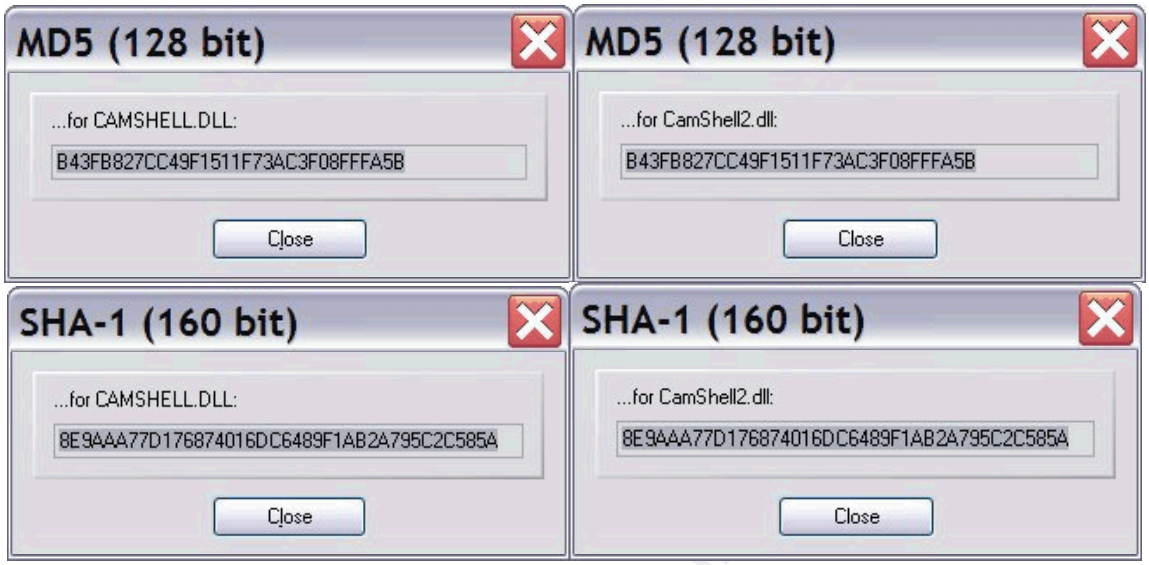


Figure 18 – MD5 and SHA-1 calculations for the Camshell.dll and Camshell2.ddl files

I can thus conclude that Mr. Leszczynski used the Camouflage program to hide 5 files inside 3 of the files contained on the floppy. These 5 files, especially the offer message shown above, prove that Mr. Leszczynski was in the process of selling Ballard's industrial secrets to another organization.

Legal Implications

With only the information contained on this floppy disk, it is not possible to prove that Mr. Leszczynski executed the Camouflage program on any system at Ballard. Although the .dll file appears on the floppy, Mr. Leszczynski could claim he used it on his home machine, and the execution of Camouflage on a private machine is not in and of itself illegal. However, with a cursory review of the system registries at Ballard (including the one assigned to Mr. Leszczynski), we should get a good idea of which systems he touched and what information he may have compromised. Additionally, other security systems present at Ballard should provide corroborating evidence of Mr. Leszczynski's actions. These include workstation and network access logging, firewall logs, and physical security measures such as badge-swipe logs and security camera records.

Depending on whether the recipient of Mr. Leszczynski's message (likely Rift, Inc. but not provable with the information on the floppy) is a foreign or domestic entity, Mr. Leszczynski has violated either Title 18 U.S.C. §1831 (Economic Espionage)[18] or Title 18 U.S.C. §1832 (Theft of Trade Secrets.)[19] The text of the two codes is similar. If Mr. Leszczynski:

- Intending or knowing that the offense will benefit any foreign government, foreign instrumentality, or foreign agent (§1831) or
- With intent to convert a trade secret, that is related to or included in a product

that is produced for or placed in interstate or foreign commerce, to the economic benefit of anyone other than the owner thereof, and intending or knowing that the offense will, injure any owner of that trade secret (§1832)

- Knowingly
  1. Steals, or without authorization appropriates, takes, carries away, or conceals, or by fraud, artifice, or deception obtains such information;
  2. Without authorization copies, duplicates, sketches, draws, photographs, downloads, uploads, alters, destroys, photocopies, replicates, transmits, delivers, sends, mails, communicates, or conveys such information;
  3. Receives, buys, or possesses such information, knowing the same to have been stolen or appropriated, obtained, or converted without authorization;
  4. Attempts to commit any offense described in paragraphs (1) through (3); or
  5. Conspires with one or more other persons to commit any offense described in paragraphs (1) through (3), and one or more of such persons do any act to effect the object of the conspiracy,

he's in for some jail time (15 years maximum under §1831, 10 years maximum under §1832) and a hefty fine ($500,000 maximum under §1831, unspecified under §1832.)

Whether Mr. Leszczynski actually executed Camouflage or any other program on a Ballard system is irrelevant. Mr. Leszczynski's "opportunity" message makes it clear he intended to benefit someone other than the owner of the trade secrets (Ballard) and his hefty asking price ensures he knew the exposure of such information would hurt Ballard. The trade secrets relate to a product produced for interstate commerce (the different state locations of Ballard's clients in the .mdb file prove that.) By mere dint of his having the confidential information on his person as he walked out the door, combined with the effort to hide it in Word documents, Mr. Leszczynski falls under the "without authorization appropriates, takes, carries away, or conceals, or by fraud, artifice, or deception obtains such information" part of item one. Claiming he did not know that the floppy and files contained proprietary information would easily be proven false by the "opportunity" letter. Given these facts, the US Government would have a good case against him (not mention a subsequent civil suit brought by Ballard against Mr. Leszczynski and his intended recipient, most likely Rift, Inc.)

Additional Information

The following sources used in my research and training should provide additional information for interested readers. Additionally, I would encourage the reader to the "Works Cited" section at the end of this document.

1. The Honeynet Project. Know Your Enemy: Learning About Security Threats, 2nd Edition. Boston: Addison-Wesley, 2004.
2. Le, Tien. Destegging tutorial. 2002. URL: http://www.unfiction.com/dev/tutorial/steg.html (4 November 2004).
3. Abzug, Mordechai T. Unofficial MD5 Page. URL: http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html (4 November 2004).

Part 2, Option 1 – Perform Forensic Analysis on a System

Verification

Synopsis of Case Facts

In order to satisfy the requirements of Part 2, Option One of the GCFA practical, I needed to analyze a possibly compromised system in an unknown state. On Tuesday, October 26, I deliberately set a system to be compromised. Such a system is commonly known as a honeypot, in that it is set up to observe the actions of an intruder as s/he cases the joint, determines how to break in, compromises the system, and sets up the system for his/her own nefarious purposes. My system was compromised less than 24 hours later. Once I had discovered the compromise, I made the decision to immediately remove power from the system to ensure the intruder did not use my system to attack other computers. Unfortunately, this limited the types of tools I could use on the system as many forensics tools for the Windows platform depend there being a "live" system for them to interact with. I could not make my system "live" again without rebooting it, the process of which would have altered information on the hard drive.

During the time I had the "honeypot" up and running, I utilized the honeypot to access the web interface on the IPCop box, and to check the operation and configuration of the "honeypot" itself. On reflection, this might not have been the best idea, as my logons and interactions were now contained within the system that I was trying to examine. However, in real life people often cannot afford the luxury of leaving an Internet-facing system completely alone, and so I worked to identify and separate the changes made by myself from those made by my assailant.

System Description

Describe the system(s) you will be analyzing

My system setup consisted of the following:

- One Gateway 2000 minitower computer, 500 mHz Intel Pentium III processor, 128MB RAM, 10/100mBps Ethernet network card, running the Windows 2000 Advanced Server operating system, to serve as the honeypot. This computer had previously been my father's personal PC; to ensure no data remained on the disk that might interfere with the investigation, I used a program called "Darik's Boot and Nuke" (DBAN)[20] which allowed me to boot the computer to a CD and wipe out the hard drive using one of several different methods. Since the only person performing a forensic examination on this computer was myself, I chose the simplest method, which writes a "0" over every portion of the hard drive:

Figure 19 – Darik's Boot and Nuke Disk Wiping Screen

I then installed Windows 2000 on a 3GB partition so that the amount of information I would have to review for my forensic examination would be relatively small. I installed Internet Information Server (IIS) version 5, as I knew that this application was very vulnerable to hacking attempts and had been the route for many intrusions on other systems. I named this computer "Bushlinks" and put up a rudimentary website with links to conservative web addresses, hoping that the angst surrounding the upcoming election would tempt an intruder who disagreed with me to come and take over my website. Such activities are commonly known as "hacktivism." To avoid arousing suspicion that this was a honeypot, I installed Microsoft Windows 2000 Service Pack 2. I hoped that an intruder would think this computer was run by an overworked administrator who didn't have time to install the latest Service Pack (4, as of this writing.) I gave this computer an internal IP address of 192.168.0.3.

- One generic-brand minitower computer, 2.4 gHz AMD Athlon, 640MB RAM, purchased new from Fry's Electronics. I erased the preinstalled operating system with DBAN, and replaced it with the Linux-based IPCop[21] operating system. The purpose of this computer was to act as an intrusion detection (but not prevention) system and "pass-through" of connections to and from the Internet by the honeypot box. I gave this computer an internal IP address of 192.168.0.1 (for communicating with the Honeypot) and let my Internet Service Provider (ISP) provide an external address, which turned out to be 69.163.89.26.
- My cable modem, connected to my Internet Service Provider (ISP).
- The Toshiba Satellite P25 laptop computer I had used in Part One, running Red Hat Linux 9 and utilizing the security tools from the GCFA class.

A diagram of the setup is shown below:

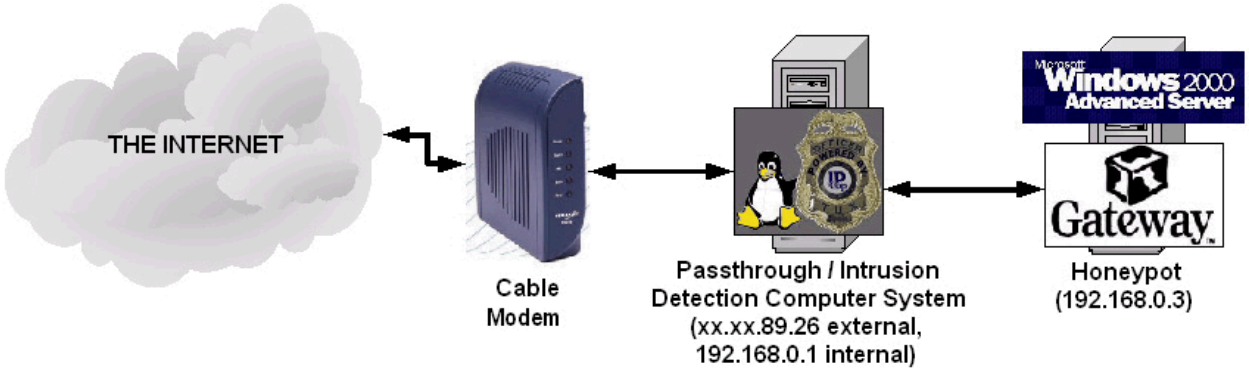
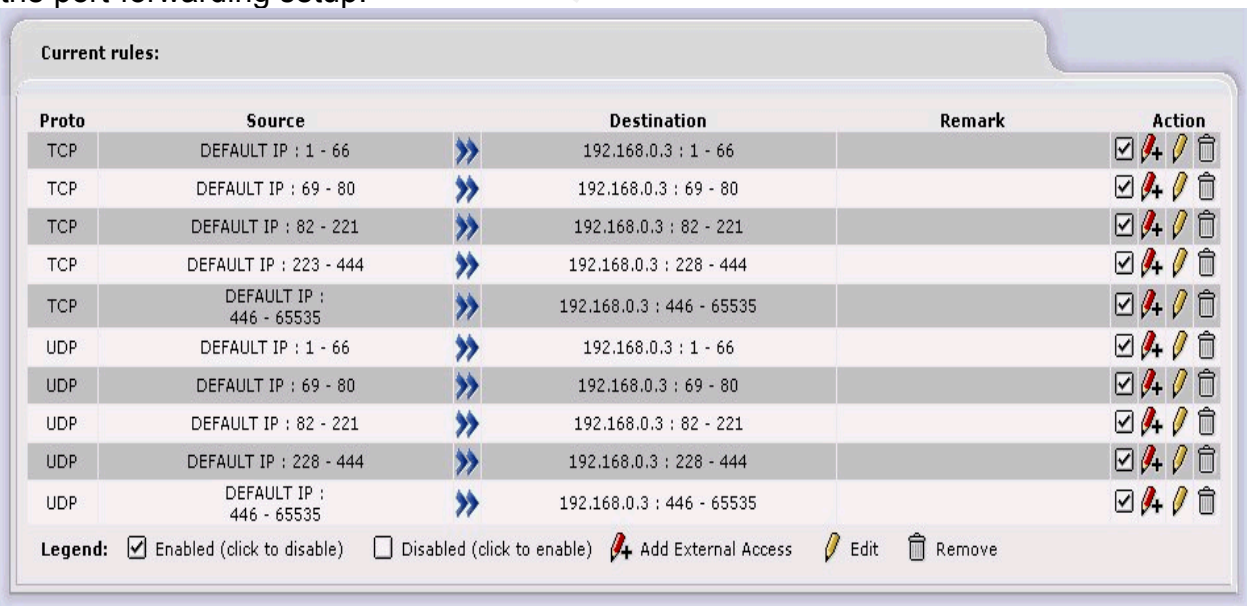


Figure 20 – Diagram of honeypot network setup

Connections from the Internet came through the cable modem, were logged (but not stopped) by the IPCop system, and finally connected to the Honeypot. IPCop normally serves to block many of the inbound connections from the Internet to a network, as these connections have the potential to cause harm to the destination computer. However, since I was deliberately trying to have my computer "hacked" I used IPCop's port-forwarding capability. Any inbound connections made to IPCop were simply forwarded along to the Honeypot; by default, any outbound connections from the honeypot were simply forwarded to the Internet. The screenshot below demonstrates the port-forwarding setup:



Current rules:

Proto	Source		Destination	Remark	Action
TCP	DEFAULT IP : 1 - 66	»	192.168.0.3 : 1 - 66		
TCP	DEFAULT IP : 69 - 80	»	192.168.0.3 : 69 - 80		
TCP	DEFAULT IP : 82 - 221	»	192.168.0.3 : 82 - 221		
TCP	DEFAULT IP : 223 - 444	»	192.168.0.3 : 228 - 444		
TCP	DEFAULT IP : 446 - 65535	»	192.168.0.3 : 446 - 65535		
UDP	DEFAULT IP : 1 - 66	»	192.168.0.3 : 1 - 66		
UDP	DEFAULT IP : 69 - 80	»	192.168.0.3 : 69 - 80		
UDP	DEFAULT IP : 82 - 221	»	192.168.0.3 : 82 - 221		
UDP	DEFAULT IP : 228 - 444	»	192.168.0.3 : 228 - 444		
UDP	DEFAULT IP : 446 - 65535	»	192.168.0.3 : 446 - 65535		

Legend: Enabled (click to disable) Disabled (click to enable) Add External Access Edit Remove

Figure 21 – IPCop Port Forwarding Rules

Note that all the rules allow connections from "DEFAULT IP" (any address, anywhere) to 192.168.0.3, the IP address of my Honeypot. Although the 192.168.0.3 address is not exposed on the Internet, anyone connecting to the external address of my passthrough / intrusion detection system (69.163.89.26) would instead see the

Honeypot.

Hardware
My evidence listing follows:

Tag Number	Description
001	Gateway 2000 minitower, PIII/500 mHz, Serial # 0016363380
002	IBM 20GB hard drive, Serial # JMYJMFB7330
003	Mitsumi CD-ROM drive, Product # FX4821T
004	CDWriter CD-R/W drive, Product # IDE5224
005	Toshiba floppy drive
006	Ensoniq Audio Controller (sound card)
007	VIA Technologies VT6105 [Rhine-III] Ethernet controller (network card)

Computer system with an IBM 20GB internal hard drive, a Mitsumi internal CD-ROM drive, a CDWriter internal CD/RW drive, a Toshiba internal high-density floppy drive, Ensoniq internal sound card, VIA internal network card. This information was discovered using the "lshw" utility on the Helix bootable CD.

Evidence Collection
Image Media
To ensure the information on the hard drive remained unaltered as I analyzed, I decided to make an exact copy (known as an "image") onto a different hard drive. However, if I were to boot the Windows 2000 operating system, it would write to the hard drive multiple times, possibly spoiling my evidence. I once again turned to the Helix bootable CD, which as noted before does not write to the host system's hard drive. Within Helix, I made use of the following utilities:

- "dd", a Linux command-line utility for making exact copies of files or even entire hard drives,
- "Netcat", commonly known as the "Swiss Army knife of network utilities", used here to send the output of "dd" over a network connection to another computer, and
- "Grab". A graphical Helix utility that automates the use of the above two programs.

I set up Grab to use dd and Netcat and asked it to calculate an MD5 fingerprint, which I would use after the transfer to compare to the MD5 fingerprint of the image on the destination computer. I also needed to set up dd and Netcat on the destination computer to receive the file, which I did at the command line:

```
[root@LinuxForensics tmp]# nc -l -p 12345 | dd of=image332.dd
```

The "nc" command is for Netcat; the "-l" option tells Netcat to listen for an incoming connection, "-p" tells it to listen on a specific port (12345, chosen arbitrarily), and the "|" (pipe) command tells the computer to send the output of the first command to the

second command. The “of” option in dd tells the computer to save its output in an output file, which I named image332.dd (I started the dd operation at 3:32 in the afternoon.)

Once the systems had been set up on both sides (sender and recipient) I activated Grab and went to get a glass of water, as the transfer process would take a long time. Once the transfer had finished, I noted the MD5 calculation created by Grab:

Figure 22 – Grab finishing the file transfer and calculating the MD5 fingerprint (“hash”)

I also confirmed that the file had been received on the Linux side, noting the match of the number of records:

```
[root@LinuxForensics tmp]# nc -l -p 12345 | dd of=image332.dd
4607019+2202594 records in
6297416+0 records out
```

The final test that the images are identical is the comparison of the MD5 fingerprints. I ran the "md5sum" utility on the image received on the Linux system:

```
[root@LinuxForensics root]# md5sum -b image332.dd
cfce5923405881675d9a195922b48d06 *image332.dd
```

We can see that the MD5 fingerprints from Grab and MD5sum exactly match, and thus conclude that the image was not altered during the file transfer process.

Operating System – Specific Media Analysis

Media Analysis of System

I used many of the same tools in my analysis of this system that I used in the floppy image analysis in Part 1. Given that the tools I had used for the floppy analysis were located on the Helix disk, given that the Helix disk did not allow access to the host machine's hard drive, and given that the size of the image (3 gigabytes) necessitated storing it on the host hard drive, I needed to use a less recent (albeit still useful) version of Autopsy, version 1.70 (the version used in the GCFA class.) To ensure Autopsy would not modify the image data as I examined it, I ran both the md5sum and sha1sum utilities before I got started and after I finished:

```
[root@LinuxForensics root]# md5sum image332.dd
cfce5923405881675d9a195922b48d06  image332.dd

[root@LinuxForensics root]# sha1sum image332.dd
05fdcc51c1659d3dff55038c0a54624654e8bfb6  image332.dd
```

A comparison of both outputs showed no alterations to the image file.

As I had installed the system myself, I knew that it used the Windows NT FileSystem (NTFS), which records more information than the FAT format. Once again, during the "import" process, Autopsy ran an MD5 calculation on my image, which I confirmed was the same as the md5sum I had just created by selecting "Image Integrity" from Autopsy's Host Manager menu:

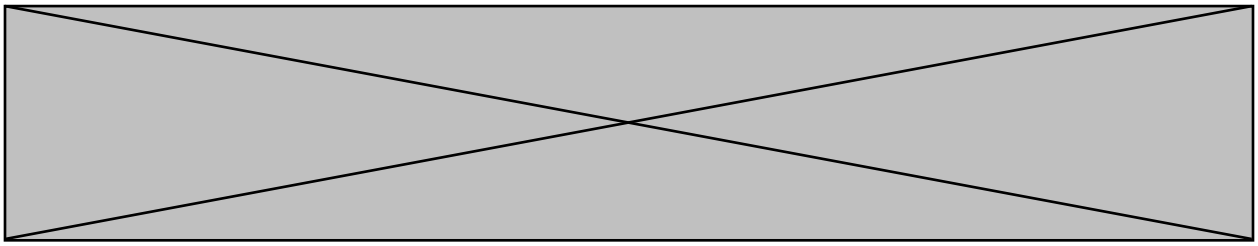


Figure 23 – Autopsy confirming MD5 fingerprint of Honeypot image prior to analysis

Although I examined various items on the image, I should note at this time that I was unable to discover the method by which my system had been compromised. Evidence of the compromise will be provided in the next section, "Timeline Analysis." However, I will review the steps I took to analyze the image.

1. File System Examination – My first step was to directly inspect the file system for evidence of modification to the operating system, "back doors" (routes that an intruder would leave in a compromised system to make his or her entry easier the next time) and "sniffer" programs (applications installed on a compromised system that examine information being passed along the network for possible use in compromising other systems.) In Autopsy, I selected "File Analysis" and visually inspected the file listings in the following directories which are typically important in a Windows 2000 installation:

Current Directory: C:\ /Documents and Settings/ /Administrator/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CHANGED	SIZE
	d / d	../	2004.10.23 23:54:33 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.23 23:54:33 (PDT)	56
	d / d	./	2004.10.23 23:54:34 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.23 23:54:34 (PDT)	344
	r / r	100_Video.asd	1999.06.29 03:12:28 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	2723
	r / r	128_CD_Transparency_Audio.asd	1999.06.30 18:22:14 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	2465
	r / r	16_AM_Radio.asd	1999.06.29 03:12:28 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	2761
	r / r	1Mb_Video.asd	1999.06.29 03:12:28 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	2730
	r / r	250_Video.asd	1999.06.29 03:12:28 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	2706
	r / r	28.8_56_100_MBR_VIDEO.asd	1999.08.27 14:27:54 (PDT)	2004.10.23 23:54:34 (PDT)	2004.10.23 23:54:34 (PDT)	4038
	r / r	28.8_FM_Radio_Mono.asd	1999.06.29	2004.10.23	2004.10.23	2564

Figure 24 – File Listing in C:\Documents and Settings\Administrator

Current Directory: C:\ /Program Files/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CHANGED	SIZE	UID	GID	META
	d / d	../	2004.10.24 15:04:14 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.26 21:41:58 (PDT)	56	48	0	5-144-6
	d / d	./	2004.10.24 00:35:59 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.24 00:35:59 (PDT)	56	0	0	3119-144-8
	d / d	Accessories/	2004.10.23 16:35:35 (PDT)	2004.10.24 12:10:48 (PDT)	2004.10.23 16:35:35 (PDT)	152	0	0	4038-144-1
	d / d	Common Files/	2004.10.24 00:33:45 (PDT)	2004.10.24 12:10:48 (PDT)	2004.10.24 00:33:45 (PDT)	56	0	0	3120-144-8
	d / d	ComPlus Applications/	2004.10.23 23:40:26 (PDT)	2004.10.24 12:10:49 (PDT)	2004.10.23 23:40:26 (PDT)	48	0	0	6562-144-1
	r / r	desktop.ini	2004.10.23 23:43:01 (PDT)	2004.10.27 22:31:19 (PDT)	2004.10.23 23:43:01 (PDT)	271	0	0	6928-128-1
	d / d	ExtractNow/	2004.10.24 00:36:01 (PDT)	2004.10.24 12:10:49 (PDT)	2004.10.24 00:36:01 (PDT)	56	0	0	6241-144-6
	r / r	folder.htt	2004.10.23 23:43:01 (PDT)	2004.10.23 23:43:01 (PDT)	2004.10.23 23:43:01 (PDT)	21952	0	0	6927-128-4
	d / d	Internet Explorer/	2004.10.23 23:42:59 (PDT)	2004.10.27 22:30:53 (PDT)	2004.10.23 23:42:59 (PDT)	56	0	0	1081-144-7
	d / d	Microsoft FrontPage/	2004.10.24 00:30:56 (PDT)	2004.10.24 12:10:49 (PDT)	2004.10.24 00:30:56 (PDT)	152	0	0	6230-144-1

Figure 25 – File Listing in C:\Program Files

Current Directory: C:\ /WINNT/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CHANGED	SIZE	UID	GID	META
	d / d	$NtServicePackUninstall$/	2004.10.24 00:21:58 (PDT)	2004.10.24 01:23:22 (PDT)	2004.10.24 00:21:58 (PDT)	168	0	0	10876-144-7
	d / d	../	2004.10.24 15:04:14 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.26 21:41:58 (PDT)	56	48	0	5-144-6
	d / d	./	2004.10.24 02:00:09 (PDT)	2004.10.27 22:30:52 (PDT)	2004.10.24 02:00:09 (PDT)	56	0	0	25-144-7
	r / r	_default.pif	1999.12.07 04:00:00 (PST)	2004.10.23 16:15:51 (PDT)	2004.10.23 16:22:32 (PDT)	707	0	0	88-128-4
	d / d	addins/	2004.10.23 16:17:02 (PDT)	2004.10.24 15:07:05 (PDT)	2004.10.23 16:22:32 (PDT)	152	0	0	62-144-1
	d / d	Application Compatibility Scripts/	2004.10.23 16:35:26 (PDT)	2004.10.24 15:07:05 (PDT)	2004.10.23 16:35:26 (PDT)	56	0	0	3914-144-7
	d / d	AppPatch/	2004.10.24 00:25:02 (PDT)	2004.10.24 15:07:05 (PDT)	2004.10.24 00:25:02 (PDT)	56	0	0	85-144-6
	r / r	Blue Lace 16.bmp	1999.12.07 04:00:00 (PST)	2004.10.23 16:35:38 (PDT)	2004.10.23 23:45:49 (PDT)	1272	0	0	4065-128-4
	r / r	certocm.log	2004.10.24 01:23:49 (PDT)	2004.10.24 01:23:49 (PDT)	2004.10.24 01:23:49 (PDT)	5915	0	0	3176-128-4
	r / r	clock.avi	1999.12.07 04:00:00 (PST)	2004.10.23 16:19:09 (PDT)	2004.10.23 23:45:49 (PDT)	82944	0	0	2670-128-4

Figure 26 – File Listing in C:\WINNT

In my examination, I did not find anything out of the ordinary. However, I would caution that since this file listing is from an image and not a running computer system, I was unable to do a complete file search.

2. Internet History Files – I made use of the program "Pasco"[22] by the security