



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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



GIAC Certified Forensic Analyst Practical assignment (GCFA) v1.5 Slade Griffin

**“I have no desire to make mysteries, but it is impossible at the moment of action to enter into long and complex explanations. “
Sherlock Holmes**

Abstract.....	3
Part 1- Analyze an unknown image.....	4
Executive Summary:	5
Examination Details.....	5
Forensic Detail and Program Identification	19
Examination Conclusion.....	31
Legal Implications.....	31
Part 2–Option 1: Perform Forensic Analysis on a System	33
Synopsis.....	33
System Details	34
Hardware Seizure	35
Image Media.....	36
Media Analysis.....	38
Timeline Analysis.....	40
Recover Deleted Files	42
Conclusion.....	44
Appendices	46
Appendix A	46
Appendix B	51
Appendix C	55
Appendix D	56
References.....	57

© SANS Institute 2005. Author retains full rights.

Abstract

This document has been completed to demonstrate the “necessary knowledge, skills, and abilities to handle advanced incident handling scenarios, conduct formal incident investigations, and carry out forensic investigation of networks and hosts.” for the GIAC Certified Forensic Analyst (GCFA) certification. The requirements for this certification are stated on the GIAC website, <http://www.giac.org/GCFA.php>.

Part one contains forensics analysis for an unknown image prepared as a report. The report outlines the steps taken, programs utilized and the functions of those programs. Part one will successfully demonstrate the candidate’s ability to perform forensic analysis, and the understanding of the forensics process and programs.

Part two of this report contains a real world example of the forensics analysis process. This example is an ongoing investigation involving over 40 major research institutions. These institutions include universities and government installations. It is an ongoing investigation that has spanned more than 10 months. This analysis will detail the candidate’s forensics analysis and involvement with this case.

© SANS Institute 2005, Author retains full rights.

Part 1- Analyze an unknown image

Ballard Industries, Inc
confidential

Memo

To: Mr. David Keen, Security Administrator

From: Slade E. Griffin, Forensic Analyst

Date: February 9, 2005

Re: Floppy #fl-260404-rjl1

Mr. Keen:

This memo is to inform you that the Information Security Office's analysis of floppy disk #fl-260404-rjl1 is complete. Forensic analysis of the disk was performed and the report is ready for you review. The report includes detailed findings of the contents and purpose of those contents on the seized floppy disk.

Please contact our department either in person or with a representative of your office to retrieve the floppy disk and chain of custody document. In the event that additional information is needed, I can be reached via voice at 4-1234 or e-mail security@ballard.com.

(enclosure)

Executive Summary:

On 26 April 2004 1645 MST, a floppy disk was seized by an authorized member of the physical security department. Mr. Robert Leszczynski, lead process control engineer, was removing the disk in question from the research and development lab. Mr. David Keen, Information Security Officer for Ballard Industries, has requested a complete analysis of the seized item. Mr. Keen explained that there has been a decline in both new and customer re-orders pertaining to the company's unique fuel cells. Internal investigations have determined that one of our rival companies has been receiving orders for the same fuel cell.

It is also a violation of company policy to remove media, floppy disks, CDs, disk drives, and hard copy from the R&D labs. Mr. Keen turned over a single floppy disk and chain of custody with the following information:

TAG# fl-260404-RJL1
3.5 inch TDK floppy disk
MD5: d7641eb4da871d980adbe4d371eda2ad (file: fl-260404-RLJ1.img)
fl-260404-RJL1.img.gz

The analysis will focus on the contents of the floppy and how Mr. Leszczynski might have used the contents.

Examination Details

Upon receipt of the floppy disk from Mr. Keen, the suspect disk was logged into custody in the forensics lab. Once the disk was logged in this room, it was placed in a locked cabinet with a key held by only two individuals in the Information Security Office. Each time the disk was removed from the locked cabinet, it will be logged out on a separate accountability sheet maintained in the cabinet itself.

Once the disk was logged in, the write protection tab was engaged to prevent data from being written to the media¹. After engaging the write protect feature of the diskette it was placed in the drive of the forensic workstation. Our forensic workstation is a dual 2.8 GHz with 6 gigabytes of RAM and 1.8 terabytes of disk space. The operating system for the unit is Hat Enterprise 3.0 update 2; the system remains air-gapped from the production network to prevent contamination.

Before mounting the image I verified the type of evidence using the "file" command. The file command is run to verify whether an object is a file or directory, and what type of file you are dealing with. The output follows:

```
# file v1_5.gz
```

v1_5.gz: x86 boot sector, system mkdosfs, FAT (12 bit)

The italicized output indicates that this is a floppy disk created with the mkdosfs command in Linux/Unix with x86 hardware. A comprehensive description of the utilization of this command can be found at the following referenceⁱⁱ. In addition, the file command reveals the type of filesystem that allowed me to mount the image in Autopsy. To maintain the integrity of the floppy disk it was mounted with the read only command as follows.

mount -r -t vfat /dev/fd0 /mnt/floppy

The command mount was used with the following instructions. The -r indicates read only, this will prevent data from being written to the file should the hardware function, mentioned previously, fail. The -t command tells the operating system which filesystem is on the media, vfat is the Linux equivalent of the dos FAT that was revealed when "file" was run. The next portion of the command is which device is being mounted. The device /dev/fd0 is the floppy drive of the forensic workstation, and the destination command of /mnt/floppy is where our evidence will be until we take an image. An image is an exact bit-by-bit copy of the original media. A complete description of the mount command is detailed at linuxvalley.itⁱⁱⁱ. Once the disk was mounted I retrieved an image of the disk to perform forensics. This keeps the disk sanitary during the analysis process. The following details how the disk was imaged once the floppy was mounted.

The main objective was to get a sanitary image of the floppy onto our forensics system. The following command was used to obtain our forensically sound image of the floppy disk. The built-in utility dd^{iv} was used to create our forensic image, the exact command follows:

dd if=/dev/fd0 of=/evidence/gcfa/v1_5.gz

The dd command is instructed to obtain the media located on /dev/fd0 as the input file, or if=. The next argument after the space is to provide the output file, or of=. In this instance, the output was stored in the directory /evidence on the forensic workstation. This is the directory on the forensics workstation where evidentiary images are stored. Once I had the image, I needed to verify that it was still intact forensically. To ensure this, I revisited the evidence tag:

TAG# fl-260404-RJL1
3.5 inch TDK floppy disk
MD5: d7641eb4da871d980adbe4d371eda2ad (file: fl-260404-RLJ1.img)
fl-260404-RJL1.img.gz

The relevant portion for this procedure is to verify the md5sum^v that is listed on the evidence tag. The md5sum is a one-way algorithm created by Professor Ronald L. Rivest of MIT. It is the de facto standard for creating a 128-bit

“fingerprint”. To verify the md5sum of the image taken against the md5sum provided by Mr. Keen the following command was issued:

#md5sum v1_5.gz

For evidentiary purposes, a screen shot was taken and the output follows.

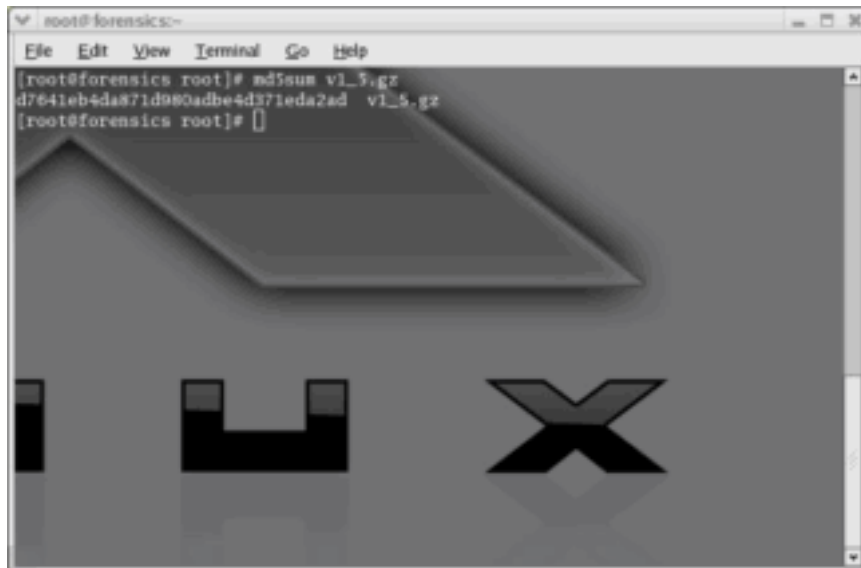


Figure 1

Figure 1 displays a good match, meaning that the image has not been altered since Mr. Keen generated the original md5sum. The md5sum of this file will be verified and noted each time the image is moved to provide a sound chain of custody. If the file were altered at any time or in any way, the md5 values would not match. For example if even one character is modified, the md5sum will change. For verification, the file command was again issued on the new image file.

#file v1_5.gz

v1_5.gz: x86 boot sector, system mkdosfs, FAT (12 bit)

The md5sum matches and the file type remains consistent. The ISO now maintains a forensically sound copy of the floppy disk. The disk was then placed and logged back in the physical evidence locker mentioned in the first paragraph of the examination details.

At this point the analysis of the created image began. I used two tools, Autopsy/Sleuthkit^{vi} and AccessData's Forensic Toolkit^{vii} (FTK). Autopsy is a free, open-source tool, and FTK was chosen for redundancy. This will allow for easy replication of results and verification if this incident is escalated into the criminal category. I utilized the two environments for verification and comparison of the

findings detailed in this report. For purposes of discussion and length only the operations of Autopsy and Sleuthkit will be discussed in the report; however evidentiary displays from FTK will be used.

The next step after image creation and verification was bringing the image into the program I wanted to use. Using Autopsy, the following steps were taken and detailed; create a new case, assign the investigators, create the host file, add an image and then create a timeline. Autopsy performed an md5sum on the image as it is imported into the program, this function verifies once again that the files has not been altered.



Figure 2

Figure 2 displayed an accurate md5sum according to the evidence tag and my own md5 check; the other displayed information was added manually as the image was imported. When I created and added the image I symlinked or moved the files. This action prevents multiple copies from being created on the forensics workstation. When the image was imported into Autopsy, some important functions are taking place besides the md5sum. Once the image was moved and the md5sum finished, I needed to create a data file. Creating a data file was done by selecting the “File Activity Timelines” from the host manager menu within Autopsy. Creating the data file was the first step to creating a timeline. I selected the floppy image, and instructed Autopsy to gather allocated, unallocated and unallocated Metadata. This tells Autopsy to collect every bit of data on the disk for analysis. The body file name field is merely a label for reference and I left it named body. I also instructed the program to generate an md5sum on the body file once it was created, this will alert me if it is tampered with. As stated before Autopsy is merely an interface for the Sleuthkit.

Here we will process the file system images, collect the temporal data, and save the data to a single file.

1. Select one or more of the following images to collect data from:

☒ a:/ images/v1_5.gz fat12

2. Select the data types to gather:

☒ Allocated Files ☒ Unallocated Files ☒ Unallocated Meta Data Structures

3. Enter name of output file (body):

output/

4. Generate MD5 Value? ☒

Figure 3

The Web browser interface is telling the Sleuthkit to run the following commands:

Running `fls -r -m on images/v1_5.gz`

Running `ils -m on images/v1_5.gz`

The command `fls` searches inode, or disk clusters and displays the data that was stored there. The switch `-r` instructs the program to look recursively through directories, or drill down. The switch `-m` is an instruction to display the output in MACtime format. The command `ils` stands for Inode Lister it will be used to look for possible deleted data, and recover it. The switch `-m` also instructs `ils` to display the output in MACtime format. MAC stands for Modified Accessed and Changed, viewing when these actions took place on a particular directory or file lends relevance to your findings by displaying when certain events took place. After the inodes were searched and the deleted data was recovered, I then created the actual timeline.

Now we will sort the data and save it to a timeline.

1. Select the data input file (body):
☒ body
2. Enter the starting date:
None: ☒
Specify: ☐ Dec 2004
3. Enter the ending date:
None: ☒
Specify: ☐ Dec 2004
4. Enter the file name to save as:
output/timeline.txt
5. Generate MD5 Value? ☒

OK

Figure 4

Figure 4 is the screen that was displayed after the body file was created this allows you to select the previously generated body file, and specific dates if you know approximately, when an incident occurred. Mr. Keen advised our office of when the disk was seized, but I did not want to perform a narrow search from the beginning so I left the start and end dates set to none. This performed timeline creation on the entire contents of the floppy image. An md5sum was also run on the output file of timeline.txt.

This page provides a monthly summary of activity.
Each day that has activity is noted with the number of events

[Feb 2001](#)
Sat Feb 03 2001 (2)

[Apr 2004](#)
Thu Apr 22 2004 (2)
Fri Apr 23 2004 (6)
Sun Apr 25 2004 (2)
Mon Apr 26 2004 (20)

Figure 5

Figure 5 displays the summary of the timeline's information. The timeline displays several dates which were reviewed newest to oldest in this report. Timelines are especially useful when given an approximate date to work from, as stated before I named the output timeline.txt here is what I found. I knew the disk was seized on 26 April 2004 1645 MST. This was the time I focused on and then moved backwards. Here is the timeline:

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

Figure 6

Figure 6 displays the timeline output in a standard text editor, instead of using the web browser window. Of particular interest are the deleted items, I took note of these immediately, and made notes in my investigator's log. The investigator's log will be available only internally, or by subpoena if necessary. Autopsy and TSK have provided valuable data already displaying items that were deleted or removed from the disk. For verification, I have also included the output from FTK. The screenshot I have inserted also details other recovered data, however we are going to note that the same files are displayed as deleted. The screenshot is broken into two separate images, figures 7 and 8, to display all available information.

File Name	Full Path	Ext	File Type	Category
Index.htm	v1_5VRJL-FAT12\Index.htm	htm	Hypertext Document	Document
Acceptable_Encryption_Policy.doc	v1_5VRJL-FAT12\Acceptable_Encryption_Policy.doc	doc	Microsoft Word XP Document	Document
CamShell.dll	v1_5VRJL-FAT12\CamShell.dll	dll	Hypertext Document	Document
DriveFreeSpace1	v1_5VRJL-FAT12\DriveFreeSpace1		Drive Free Space	Slack/Free Space
FAT1	v1_5VRJL-FAT12\FAT1		File Allocation Table	Slack/Free Space
FAT2	v1_5VRJL-FAT12\FAT2		File Allocation Table	Slack/Free Space
FileSystemSlack	v1_5VRJL-FAT12\FileSystemSlack		File System Slack	Slack/Free Space
Information_Sensitivity_Policy.doc	v1_5VRJL-FAT12\Information_Sensitivity_Policy.doc	doc	Microsoft Word XP Document	Document
Internal_Lab_Security_Policy.doc	v1_5VRJL-FAT12\Internal_Lab_Security_Policy.doc	doc	Microsoft Word XP Document	Document
Internal_Lab_Security_Policy1.doc	v1_5VRJL-FAT12\Internal_Lab_Security_Policy1.doc	doc	Microsoft Word XP Document	Document
Password_Policy.doc	v1_5VRJL-FAT12>Password_Policy.doc	doc	Microsoft Word XP Document	Document
Remote_Access_Policy.doc	v1_5VRJL-FAT12\Remote_Access_Policy.doc	doc	Microsoft Word XP Document	Document
VBR	v1_5VRJL-FAT12\VBR		Volume Boot Record	Slack/Free Space

Figure 7

Figure 6 is the first half of the screen after importing the image. In addition to confirming the deleted files and MACtimes, FTK is displaying that CamShell.dll is a hypertext document rather than a dll. In the MS-Windows environment a dll is defined as “a file containing a collection of Windows functions designed to perform a specific class of operations. Most DLLs carry the .DLL extension, but some Windows DLLs, such as Gdi32.exe, use the .EXE extension. Functions within DLLs are called (invoked) by applications as necessary to perform the desired operation.”^{viii} This definition states that a dll is used to run a program, further investigation will be needed on the specific file to determine what the program identity and function are. It has automatically recovered slack space, displays relevant file system information, recovers deleted files where it can, and it will also check extensions. Slack space is the space between the end of a file and the end of the cluster where the file sits. Slack space may contain the remnants of deleted files so recovering it can be very useful during an investigation. This is a comprehensive list of all files in the image. AccessData maintains a database of “Known File Formats”, or a KFF database. It looks through files to see if the extensions have been altered. This is why CamShell.dll is in red.

Cr Date	Mod Date	Acc Date	L-Size	P-Size	Chi...	De...	Enc	Del	R...	Idx	Sector	Cluster
4/26/2004 9:47:36 AM	4/23/2004 10:53:56 ...	4/26/2004	727	1,024	0	0				Full	33	2
4/26/2004 9:46:44 AM	4/23/2004 2:10:50 PM	4/26/2004	22,528	22,528	5	5				Full		
4/26/2004 9:46:18 AM	2/3/2001 7:44:16 PM	4/26/2004	36,864	36,864	0	0		Y		Full	33	2
N/A	N/A	N/A	798,208	26,214,400	0	0				Full		
N/A	N/A	N/A	4,608	4,608	0	0				Full	1	
N/A	N/A	N/A	4,608	4,608	0	0				Full	10	
N/A	N/A	N/A	4,096	4,096	0	0				Full	2,872	
4/26/2004 9:46:20 AM	4/23/2004 2:11:10 PM	4/26/2004	42,496	42,496	5	5				Full		
4/26/2004 9:46:24 AM	4/22/2004 4:31:06 PM	4/26/2004	33,423	33,792	5	5				Full		
4/26/2004 9:46:22 AM	4/22/2004 4:31:06 PM	4/26/2004	32,256	32,256	5	5				Full		
4/26/2004 9:46:26 AM	4/23/2004 11:55:26 ...	4/26/2004	307,935	308,224	5	5				Full		
4/26/2004 9:46:36 AM	4/23/2004 11:54:32 ...	4/26/2004	215,895	216,064	5	5				Full		
N/A	N/A	N/A	512	512	0	0				Full	0	

Figure 8

Two files were deleted on the same day the disk was seized, and one appears to have been altered according to the KFF maintained by FTK. Another suspicious fact displayed in Figures 6 and 7 are extremely large file sizes of at least two files which appear to be MS-Word Documents. These file sizes were the next note in my investigator’s log. Having completed the timeline verification with FTK, and also having seen some new information, I then proceeded with the investigation

[illegible]

Figure 8 is the User interface displaying graphically the contents of our floppy image based on the creation of the body and timeline files. I noted that dates, times, and files matched up with FTK's output. Starting from the top of this screen, I first reviewed _ndex.htm. The first character being replaced with the underscore is indicative of the Microsoft Windows deletion process, the contents of the html code are displayed below:

This is standard HTML code on a web page named "ballard" with an embedded ShockWave Flash movie. It does not display any particularly useful information; I did note the file's size and MACtimes as I proceed to the next deleted file. The second file was identical to index.htm yet appeared to be quite a bit larger. To

#	File	Offset	Size	SHA1	SHA256	MD5	MD6	MD8	MD10	MD12	MD14	MD16	MD18	MD20	MD22	MD24	MD26	MD28	MD30	MD32	MD34	MD36	MD38	MD40	MD42	MD44	MD46	MD48	MD50	MD52	MD54	MD56	MD58	MD60	MD62	MD64	MD66	MD68	MD70	MD72	MD74	MD76	MD78	MD80	MD82	MD84	MD86	MD88	MD90	MD92	MD94	MD96	MD98	MD100	MD102	MD104	MD106	MD108	MD110	MD112	MD114	MD116	MD118	MD120	MD122	MD124	MD126	MD128	MD130	MD132	MD134	MD136	MD138	MD140	MD142	MD144	MD146	MD148	MD150	MD152	MD154	MD156	MD158	MD160	MD162	MD164	MD166	MD168	MD170	MD172	MD174	MD176	MD178	MD180	MD182	MD184	MD186	MD188	MD190	MD192	MD194	MD196	MD198	MD200	MD202	MD204	MD206	MD208	MD210	MD212	MD214	MD216	MD218	MD220	MD222	MD224	MD226	MD228	MD230	MD232	MD234	MD236	MD238	MD240	MD242	MD244	MD246	MD248	MD250	MD252	MD254	MD256	MD258	MD260	MD262	MD264	MD266	MD268	MD270	MD272	MD274	MD276	MD278	MD280	MD282	MD284	MD286	MD288	MD290	MD292	MD294	MD296	MD298	MD300	MD302	MD304	MD306	MD308	MD310	MD312	MD314	MD316	MD318	MD320	MD322	MD324	MD326	MD328	MD330	MD332	MD334	MD336	MD338	MD340	MD342	MD344	MD346	MD348	MD350	MD352	MD354	MD356	MD358	MD360	MD362	MD364	MD366	MD368	MD370	MD372	MD374	MD376	MD378	MD380	MD382	MD384	MD386	MD388	MD390	MD392	MD394	MD396	MD398	MD400	MD402	MD404	MD406	MD408	MD410	MD412	MD414	MD416	MD418	MD420	MD422	MD424	MD426	MD428	MD430	MD432	MD434	MD436	MD438	MD440	MD442	MD444	MD446	MD448	MD450	MD452	MD454	MD456	MD458	MD460	MD462	MD464	MD466	MD468	MD470	MD472	MD474	MD476	MD478	MD480	MD482	MD484	MD486	MD488	MD490	MD492	MD494	MD496	MD498	MD500	MD502	MD504	MD506	MD508	MD510	MD512	MD514	MD516	MD518	MD520	MD522	MD524	MD526	MD528	MD530	MD532	MD534	MD536	MD538	MD540	MD542	MD544	MD546	MD548	MD550	MD552	MD554	MD556	MD558	MD560	MD562	MD564	MD566	MD568	MD570	MD572	MD574	MD576	MD578	MD580	MD582	MD584	MD586	MD588	MD590	MD592	MD594	MD596	MD598	MD600	MD602	MD604	MD606	MD608	MD610	MD612	MD614	MD616	MD618	MD620	MD622	MD624	MD626	MD628	MD630	MD632	MD634	MD636	MD638	MD640	MD642	MD644	MD646	MD648	MD650	MD652	MD654	MD656	MD658	MD660	MD662	MD664	MD666	MD668	MD670	MD672	MD674	MD676	MD678	MD680	MD682	MD684	MD686	MD688	MD690	MD692	MD694	MD696	MD698	MD700	MD702	MD704	MD706	MD708	MD710	MD712	MD714	MD716	MD718	MD720	MD722	MD724	MD726	MD728	MD730	MD732	MD734	MD736	MD738	MD740	MD742	MD744	MD746	MD748	MD750	MD752	MD754	MD756	MD758	MD760	MD762	MD764	MD766	MD768	MD770	MD772	MD774	MD776	MD778	MD780	MD782	MD784	MD786	MD788	MD790	MD792	MD794	MD796	MD798	MD800	MD802	MD804	MD806	MD808	MD810	MD812	MD814	MD816	MD818	MD820
---	------	--------	------	------	--------	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

For verification, the output from FTK for the file camshell.dll is displayed:

00000000	3c 48 54 4d 4c 3e 0d 0a-3c 48 45 41 44 3e 0d 0a	<HTML>...<HEAD>...
00000010	3c 6d 65 74 61 20 68 74-74 70 2d 65 71 75 69 76	<meta http-equiv
00000020	3d 43 6f 6e 74 65 6e 74-2d 54 79 70 65 20 63 6f	=Content-Type co
00000030	6e 74 65 6e 74 3d 22 74-65 78 74 2f 68 74 6d 6c	ntent="text/html
00000040	3b 20 20 63 68 61 72 73-65 74 3d 49 53 4f 2d 38	; charset=ISO-8
00000050	38 35 39 2d 31 22 3e 0d-0a 3c 54 49 54 4c 45 3e	859-1">...<TITLE>
00000060	42 61 6c 6c 61 72 64 3c-2f 54 49 54 4c 45 3e 0d	Ballard</TITLE>..
00000070	0a 3c 2f 48 45 41 44 3e-0d 0a 3c 42 4f 44 59 20	..</HEAD>...<BODY
00000080	62 67 63 6f 6c 6f 72 3d-22 23 45 44 45 44 45 44	bgcolor="#EDED
00000090	22 3e 0d 0a 0d 0a 3c 63-65 6e 74 65 72 3e 0d 0a	">...<center>..
000000a0	3c 4f 42 4a 45 43 54 20-63 6c 61 73 73 69 64 3d	<OBJECT classid=
000000b0	22 63 6c 73 69 64 3a 44-32 37 43 44 42 36 45 2d	"clsid:D27CDB6E-
000000c0	41 45 36 44 2d 31 31 63-66 2d 39 36 42 38 2d 34	AE6D-11cf-96B8-4
000000d0	34 34 35 35 33 35 34 30-30 30 30 22 0d 0a 20 63	44553540000"... c
000000e0	6f 64 65 62 61 73 65 3d-22 68 74 74 70 3a 2f 2f	odebase="http://
000000f0	64 6f 77 6e 6c 6f 61 64-2e 6d 61 63 72 6f 6d 65	download.macrome
00000100	64 69 61 2e 63 6f 6d 2f-70 75 62 2f 73 68 6f 63	dia.com/pub/shoc
00000110	6b 77 61 76 65 2f 63 61-62 73 2f 66 6c 61 73 68	kwave/cabs/flash
00000120	2f 73 77 66 6c 61 73 68-2e 63 61 62 23 76 65 72	/swflash.cab#ver
00000130	73 69 6f 6e 3d 36 2c 30-2c 30 2c 30 22 0d 0a 20	sion=6,0,0,0"...
00000140	57 49 44 54 48 3d 22 38-30 30 22 20 48 45 49 47	WIDTH="800" HEIG
00000150	48 54 3d 22 36 30 30 22-20 69 64 3d 22 62 61 6c	HT="600" id="bal
00000160	6c 61 72 64 22 20 41 4c-49 47 4e 3d 22 22 3e 0d	lard" ALIGN="">..
00000170	0a 20 3c 50 41 52 41 4d-20 4e 41 4d 45 3d 6d 6f	.. <PARAM NAME=mo
00000180	76 69 65 20 56 41 4c 55-45 3d 22 62 61 6c 6c 61	vie VALUE="balla
00000190	72 64 2e 73 77 66 22 3e-20 3c 50 41 52 41 4d 20	rd.swf"> <PARAM

Figure 11

This is the exact same code that was seen in the deleted _index.html file displayed in Autopsy. Furthermore, the output of the dll instructions is also identical. To further review the dll instructions, I have excerpted the relevant pieces. These pieces allowed me to use the internet as a resource to discover the identity and purpose of the program that used the dll. This will be my keyword, or dirty word list.

```

11\SheCamouflageShell
VB5! .
\AC:\My Documents\VB
Programs\Camouflage\Shell\CamouflageShell.vbp

CamouflageShell NewFolder RegOpenKeyEx
ExplorerNameCamouflage ExplorerNameUncamouflage
C:\WINDOWS\SYSTEM\MSVBVM60.DLL\3
Camouflage.exe http://www.camouflage.co.uk

```

Figure 12

This was enough of a foundation to begin researching this piece of software. The camshell.dll was moved to a sterile system in the forensics lab and I attempted to execute or open the .html file. Observing the running processes and md5sums of critical operating system binaries it did not appear to cause any problems or execute any programs. This html code being identical in each file, the relevant pieces of the dll, and the exportation of the file were noted in my log.

I now referred back to the timeline for comparison of the dll files code, and the deletion of the aforementioned files.

These are pieces of what appear to be a Visual Basic script or program that was written in VB according to our timeline at 0946 on 4/26/04. Looking at the deleted _ndex.htm file I see its creation time as 0947 on the same day. I went to Google^{ix} to investigate MSVBVM60.DLL\3. I am going to assume that the \3 is a further instruction and just look up the file. Below is an interesting description from liutilities^x:

DLL File: msvbvm60 or msvbvm60.dll
DLL Name: VB Virtual Machine

Description:
msvbvm60.dll is a module for the Microsoft Visual Basic virtual machine.

Part Of: Visual Basic

System DLL: No

Common Errors: File Not Found, Missing File, Exception Errors

The output from liutilities confirms the program was coded with Visual Basic. Microsoft, In their quest to make things more accessible to everyone, sometimes automates or scripts functions within their platform. This automation can have security implications. I also used Google to research the \3 instruction at the end of the msbvm execution. I got many references to sluggish systems and spyware cleaning. Not seeing what I needed from Google, I questioned an application developer. Here is what he says

“u can put it in ur website and it will install itself.. looks like a vb script.. camoflages(spelled wrong) itself and hides in user home directory (like mydocuments).. it uses the VB file "C:\WINDOWS\SYSTEM\MSVBVM60.DLL\3", the \3 is 'numeric tail' -> this slows the system down.. basically to disarm the anti-virus agent.. list of dll's that it uses:

ole32.dll - normal

shell32.dll - normal

advapi32.dll - normal

CamShell.dl - makes it invisible

“

After further investigation, I also believe the \3 to be a normal instruction during the install of a program that will make system calls to certain dll files. Past experience locating similar deleted files and now, discovering the camshell.dll code was simply hidden under the padding of the HTML code led me back to Google to discover the purpose of the program and its identity. If a dll was intentionally hidden under some HTML, this would allow someone to get the code in and out of the building had the physical security agents not been alert.

Regardless of the reason for placing the html over the top of the dll. discovering the program's identity and purpose became paramount.

Using the terms extracted in figure 11, I began to research the possible meaning of camshell.dll. I searched Google for camshell.dll, and received the following hits.

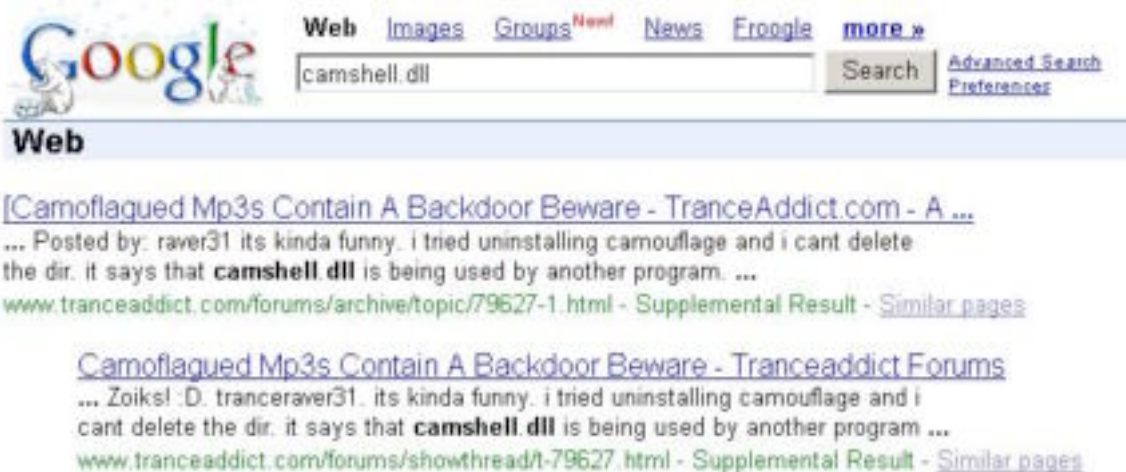


Figure 13

Research conducted on those web pages led to a further discovery of the actual software and its purpose. This website was a web forum where users were discussing the use of a piece of software called Camouflage. The word camouflage was used many times in figure 11, and I considered this to be no coincidence. The web link in figure 11 was no longer active, further research into the code revealed in figure 9 yielded additional information that also proved useful. I noted that there appears to be extensive garbled output at the bottom of figure 9. At this point I decide to open the suspect file, camshell.dll, in a hexadecimal editor. Hex editors as they are often called, allow you to more easily view data that may not normally human readable. A hex editor can sometimes help you view previously unreadable data. Below is the non-readable portion at the end of figure 9. Expressed in the contents are several clues that enabled me to determine the identity, and later the purpose, of the program that uses camshell.dll. These clues were pivotal pieces evidence during the forensic analysis.

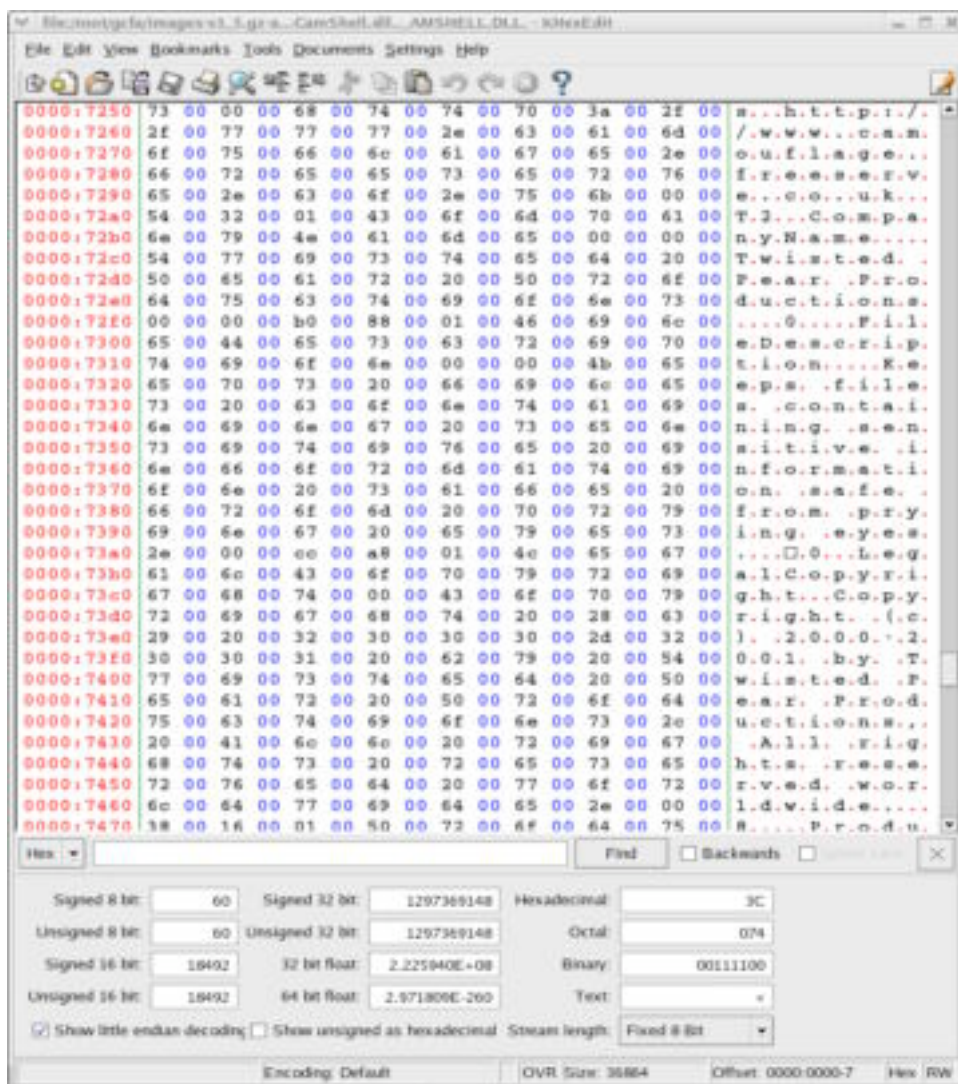


Figure 14

Figure 14 displays a web address and the copyright information for this piece of software. The web address was no longer available but the words “camouflage” and “twisted pear productions” were vital. I again used Google to extract a more legible shot of the text revealed by the hex editor. This search revealed numerous references to the author of the Camouflage software, and the software’s intended purpose. In addition I was able to obtain a working copy of the software which I downloaded to a non-forensics workstation in the lab. The definition and purpose of Camouflage are listed below

Hide files from prying eyes
Derek Mason, Twisted Pear Productions

Camouflage v1.0.4 - These days companies are given more power to monitor e-mails and to examine your personal files. And with more and more malicious "spy" software being widely used, you need to be sure that files containing sensitive information are kept safe from prying eyes. Electronic privacy is no longer guaranteed - who knows who

might be intercepting your e-mails or scanning your hard drive without your knowledge or consent? But now you can "camouflage" your sensitive files to prevent unauthorized discovery. E-mail anything you like without your attached files being revealed! **Hide your most secret files with Camouflage.** Camouflage allows you to hide files by scrambling them and then attaching them to the file of your choice. This camouflaged file then looks and behaves like a normal file, and can be stored, used or e-mailed without attracting attention. For example, you could create a picture file that looks and behaves exactly like any other picture file but contains hidden encrypted files, **or you could hide a file inside a Word document that would not attract attention if discovered.** Such files can later be safely extracted. For additional security you can password your camouflaged file. This password will be required when extracting the files within. You can even camouflage files within camouflaged files. Camouflage was written for use with Win95/98/ME/NT/2000, and is extremely easy to install and use; 1399149 bytes

Figure 15

Figure 15 is describing a process called steganography. Steganography is defined as "The practice of hiding one piece of information inside of another. The most common example is watermarking." ^{xi}. Mr. Leszczynski is the lead process control engineer and has access to many different types of data. I will focus the remainder of my forensic efforts here as I try to reveal the use of this piece of software. In addition, I will attempt to prove that this is the exact same piece of software used.

Forensic Detail and Program Identification

Google revealed that a current page containing Camouflage could be found here: <http://camouflage.unfiction.com/>. This web address was opened and the main page states that it is maintained by the author of the Camouflage software. There do seem to be legitimate uses for the software such as digital watermarking. I downloaded and installed the product on a non-sterile workstation that does not interact with our forensics workstations. Camouflage is an easy to use, powerful, and dangerous steganography tool. Steganography has almost no legitimate use within the confines of the Ballard industries network. I took this opportunity to successfully "steg" a picture within another file. Beyond installing itself like a normal program, it embeds itself in the Windows shell allowing right-click functionality and has a simple wizard GUI to step you through the use of this simple piece of software. In order to determine if this software was used, I performed forensic verification of a file I noticed on the non-sterile workstation.



Figure 16

The camshell.dll, and the program path match the forensic output displayed previously. In addition to having the same name, the timestamp matches the time and date given in figures 7 and 8. Having a similarly named file, and the same program path were not enough to establish that this is the exact same program. In order to prove this I needed to verify using md5sum. In order to achieve this I needed to check the camshell.dll that was just installed, against the camshell.dll that had the HTML code placed on top of it. The first step was to view both in a hex editor since I knew that the camshell.dll that was seized had been altered. Using a forensics CD I ran md5sum against the camshell.dll that was installed on our non-sterile workstation. I used static binaries that are stored on a forensics CD named Helix^{xii}, this useful collection of utilities is maintained by Drew Fahey.

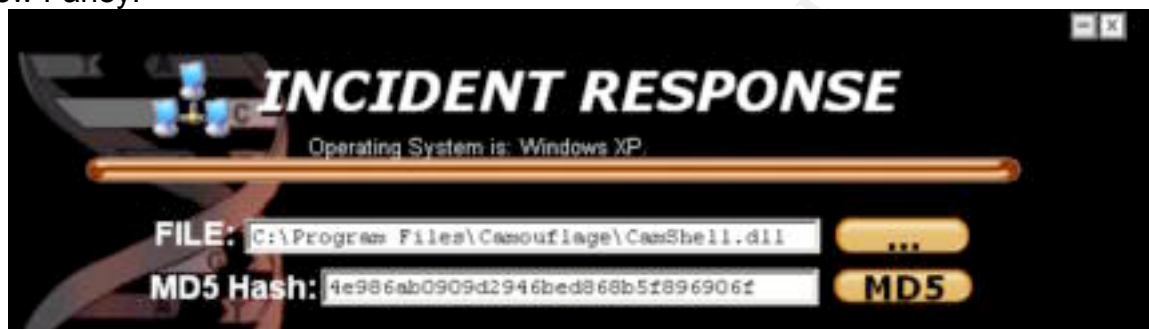


Figure 17

Figure 16 is the first of four different md5sums I will perform during the verification process. With an md5 created it was now safe to move this file to my sterile Linux workstation for further verification. The file was placed on a previously unused floppy and copied to the forensics workstation. Now that I have them both on the same system, I wanted to view them both in a hex editor side by side. This will allow me to prove not only that both files are identical, how one was altered, and that this is without a doubt the exact same program. Because the _ndex.htm file output and the beginning of the camshell.dll are identical I first want to redisplay that side by side. Figure 17 below shows the code is indeed identical.

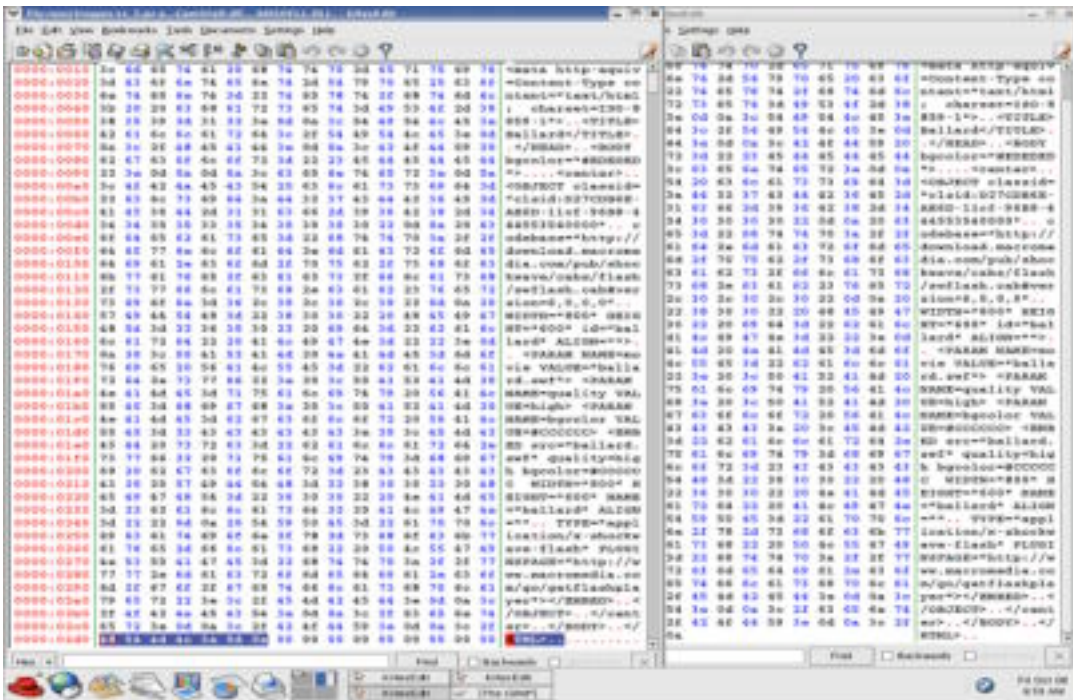


Figure 18

Figure 18 displays that they are identical until the very end, where camshell.dll, on the left, keeps going. What I will do here is copy from the highlighted portion of camshell.dll back to the beginning of the file. This removal represents the entire contents of _index.htm, which is displayed on the right. I am displaying that this amount of the file is what overwrote the original code which should indeed give us a good md5. I then performed an md5sum on the seized camshell.dll. First I exported the file from autopsy to the /root directory on the forensics workstation and named verify1.

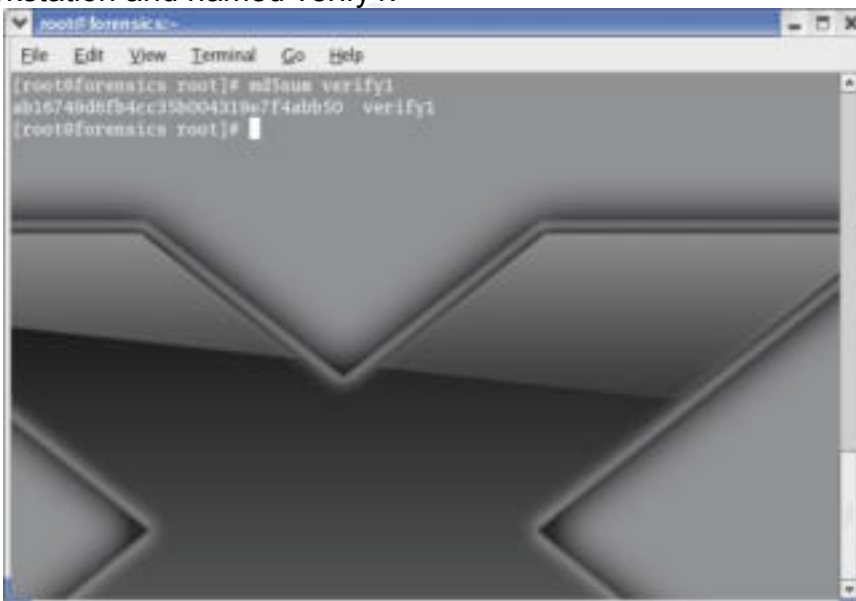


Figure 19

Figure 19's md5sum does not match the md5sum taken on our windows machine. Next, I looked at the camshell.dll that was installed on the windows workstation in a hex editor.

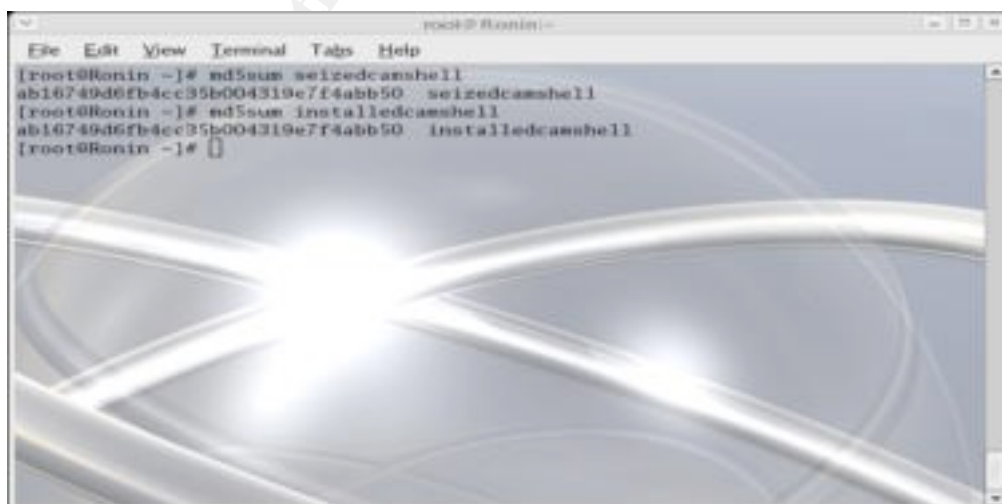
```

MZ|.....ÿÿ.....@.....
|.....E.....
|..%..!|..!|This program cannot
|be run in DOS mode...$.....
|xNUo. |o. |o. |i. |n. |;) |k. |
|Â;-|n. |!9$|n. |Richo. |.....
|.....PE..L..|_|:.....Â..!
|.....P...0.....X.....7
|.....|.....
|.....Z..*...DS..(.....p..x.....
|.....|.....
|.....|.....
|.....text...*J.....P.....
|.....data.....
|.....@..Â.rsrc...x...p...
|.....p.....@...@.reloc...
|.....|.....@...B
|Âm/9.....MSVBVM60.DLL.....

```

Figure 20

Figure 20 is the header from the camshell.dll that we installed on our windows machine. Viewing the _ndex.htm in a hex editor revealed that the HTML code ended at hexadecimal offset 0000:02d6. Knowing the stopping point of the original HTML code allows me to remove that amount of data from the recovered camshell.dll, floppy image, and compare it to the created camshell.dll, camouflage install. I removed all data prior to 0000:02d6 from the camshell.dll I recovered from the seized floppy. This file was named seizedcamshell. This data is the HTML code previously displayed in figure 17. I then opened the camshell.dll created by Camouflage during its install and performed the same procedure, removing all data prior to 0000:02d6. This wipes out the exact same amount of data. This file was named installedcamshell. With the files now saved, I performed an md5 fingerprint.



```

root@Ronin:~# md5sum seizedcamshell
ab16749d6fb4cc35b004319e7f4abb50  seizedcamshell
root@Ronin:~# md5sum installedcamshell
ab16749d6fb4cc35b004319e7f4abb50  installedcamshell
root@Ronin:~#

```

Figure 21

Figure 21 shows that if that exact amount of data is removed then the files are identical. For further verification I inserted the data prior to offset 0000:02d6 from the seized camshell.dll into the now blank header of installedcamshell. I named the original seized camshell with the html code intact reccam, and the installed camshell.dll with the inserted HTML code installedcamshell1.



```
root@Ronin:-  
File Edit View Terminal Tabs Help  
[root@Ronin ~]# md5sum reccam  
6462fb3acca0301e52fc4ffa4ea5eff8 reccam  
[root@Ronin ~]# md5sum installedcamshell1  
6462fb3acca0301e52fc4ffa4ea5eff8 installedcamshell1  
[root@Ronin ~]#
```

Figure 22

Figures 21 and 22 successfully demonstrate that camshell.dll from the floppy image, and camshell.dll from the installed software are identical. Beyond the md5 fingerprint the timeline activity from figures 9 and install information from figure 15 both display identical dates and times. The identity of the program has been revealed, and earlier I discussed the program's ease of use in hiding a file within another file. To determine whether or not it was used I decided to look more closely at the documents from the seized floppy. I needed to look closely at the documents contained on the seized floppy's image. At this point it became necessary to run **strings** against every file located on the disk. The relevant discovery and output follows.

I started with the first file displayed in the forensic browser displayed in figure 8. These documents appear to be copies of several internal Ballard industries policies. The list of files on the floppy image is as follows:

1. _ndex.html (deleted)
2. Acceptable_Encryption_Policy.doc (intact)
3. CamShell.dll (deleted)
4. Information_Sensitivity_Policy.doc (intact)
5. Internal_Lab_Security_Policy.doc (intact)
6. Internal_Lab_Security_Policy1.doc (intact)

7. Password_Policy.doc (intact)
8. Remote_Acess_Policy.doc (intact)

Files 1 and 3 have been sufficiently analyzed, the remainder of my forensic effort will focus on the other documents contained on the seized floppy. Files 2 and 4 display no unusual attributes when viewed in either their native format, Microsoft-Word, or when strings is run against them. Document 5 displayed extended characters after the normal MS-Office end of file registration tags. Document 6, which has a remarkably similar name, has no such extended characters. Documents 7 and 8 displayed significantly more extended characters, in addition to having file sizes that are disproportionate for a text document of that length. A sample of the extended characters follows:

```
Microsoft Word 10.0
Ballard
Cisco Systems, Inc.
Remote Access Policy
Title
Microsoft Word Document
MSWordDoc
Word.Document.8
Remote Access Policy
Normal.dot
Microsoft Word 10.0
Ballard
O6pQ
gW^b!
)AqXSh]
" LJ
\N> )x)
NNVs
ZKfqjj
(J$?
nQh`n
;ptj`
```

Figure 23

Knowing the purpose of the Camouflage program and having found extended data appended to three of the recovered documents, it is reasonable to assume that Camouflage was used to embed data inside these files. In order to prove this I exported the files from the floppy image and moved them to the workstation where I installed Camouflage. Before moving the files, an md5sum was performed, and the non-sterile workstation was air-gapped from the Ballard network. Once the files were copied to the Windows workstation, md5 verification was performed. Being only somewhat familiar with the operation of the program, I performed some research on the use of steganography tools and, in particular, the use of Camouflage. I found an excellent article^{xiii} that described the software and its use in detail. Following my research I proceeded to uncamouflage the suspect files

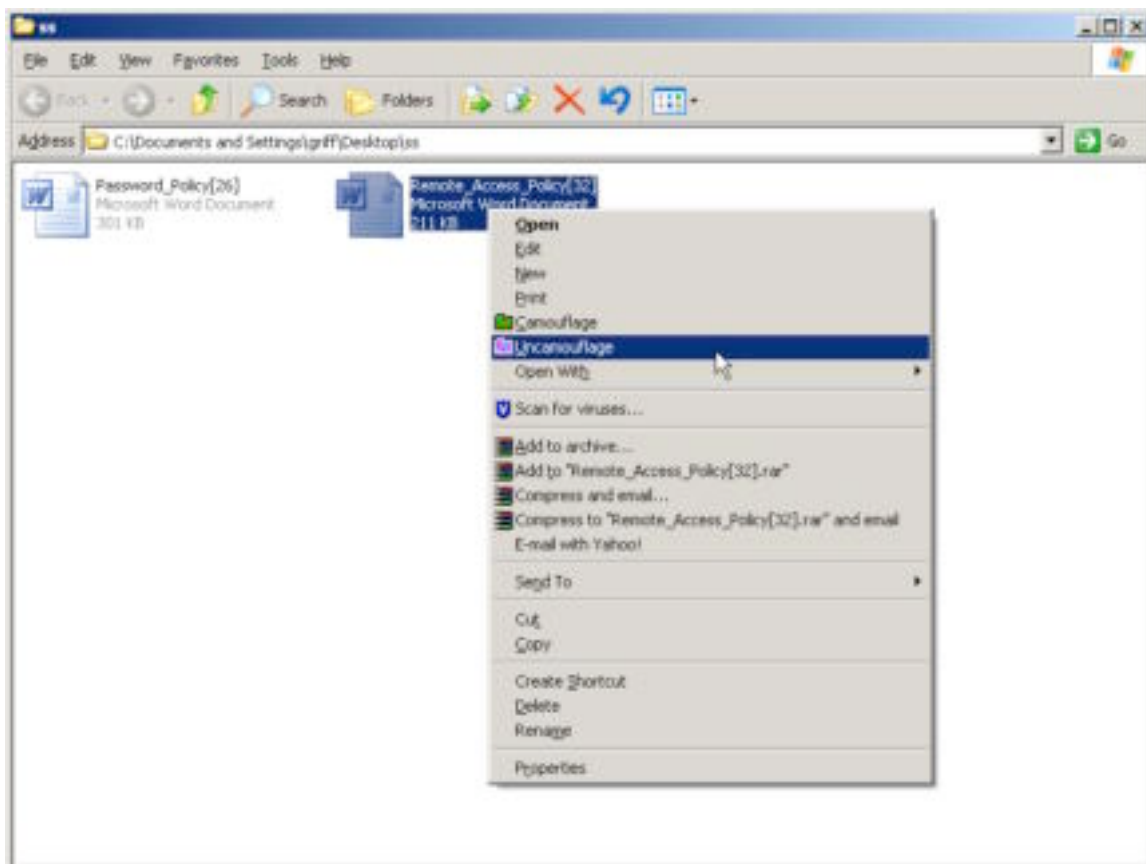


Figure 24

Figure 24 displays the two documents with extremely large file sizes. Performing a right-click and selecting Uncamouflage will display the following dialog box.

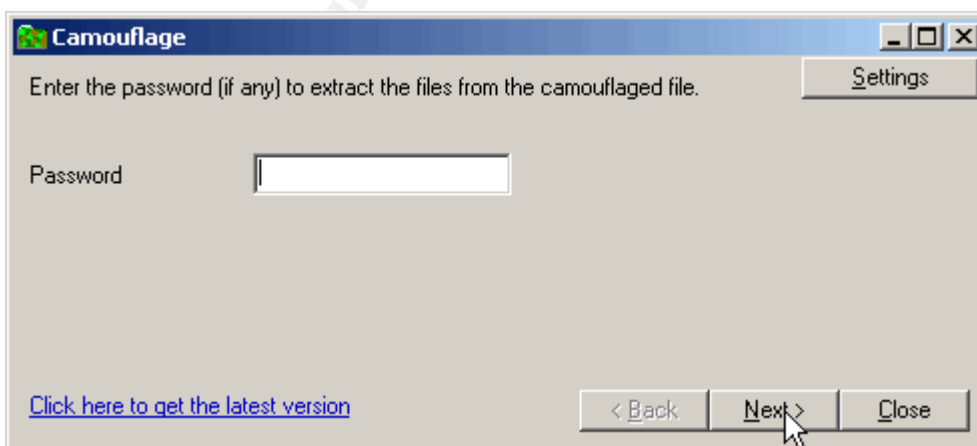


Figure 25

Figure 25 is displayed whether or not there is an actual password according to the research I performed.

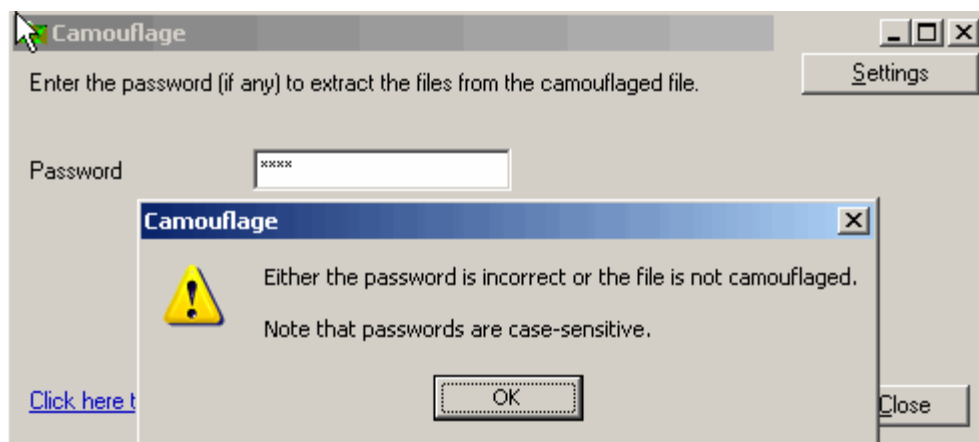


Figure 26

Figure 26 demonstrates my attempts at guessing the password incorrectly. At this point I referred to the frequently asked questions, FAQ, portion of the author's web site.

10. I've forgotten my password and can't uncamouflage a file. What can I do?

Camouflage always asks you for a password whether the file is camouflaged or not, or whether it is a camouflaged file with a password or not. This is because Camouflage doesn't give the game away that a file may be camouflaged. For security reasons we cannot release a program to reveal passwords in camouflaged files. If you forget your password we can't usually help you.

Be careful when typing in passwords - check your CAPS LOCK because Camouflage passwords are case-sensitive.

The Author is stating that the password is encrypted and that they probably cannot help. Proprietary encryption is usually fairly simple, I Camouflaged two unrelated files and examined how the program stores and "encrypts the password. I used no password on one file, and viewed the output with a hex editor. I then did the same thing with the password aaa, and viewed the output. The next step was to increment the password, by either character or number of characters, e.g. aab or aaaa. Having done this I see that it stores the password in the same place every time but it is also definitely scrambled or "encrypted." Many companies that make up their own encryption do very simple bit inversion, or sometimes they will think of a simple algorithm like +3 -2 +4 and just keep following that.

Here is a shot of one or two increments.



Figure 27

The left was when I used a and the right was when b was used. Performing this numerous times with different permutations revealed the encryption algorithm. The Algorithm is attached as Appendix A. I decided to take a closer look at all of the documents to make sure I had not missed anything. Looking at our main screen in Autopsy tells me a couple of things about two very similar files. Here is the screen:

r/s	Internal_Lab_Security_Policy.doc (D87B8-2.30K)	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/s	Internal_Lab_Security_Policy1.doc (D87B8-1.30K)	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

Figure 28

Figure 28 shows that they were written to the disk within two seconds of each other, but Microsoft's file naming abbreviation shows that Policy1 was actually the original file. It is also obvious that the newer file is slightly bigger than the original. I decided to see if it was camouflaged the same way. Viewing this document in a hex editor, and comparing it with the broken encryption algorithm revealed this file used no password.

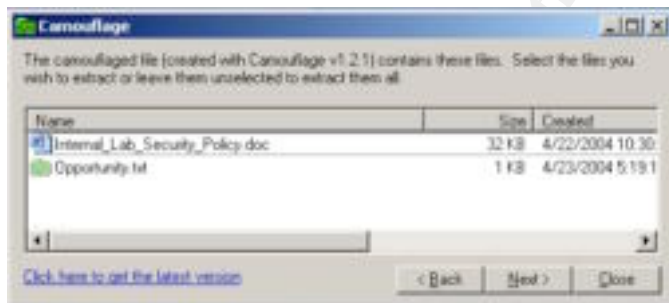


Figure 29

Figure 29 was revealed when a blank password was used. Opportunity.txt was embedded within the document. Clicking next produced the following dialog box, where I chose the directory to save the embedded or hidden file.

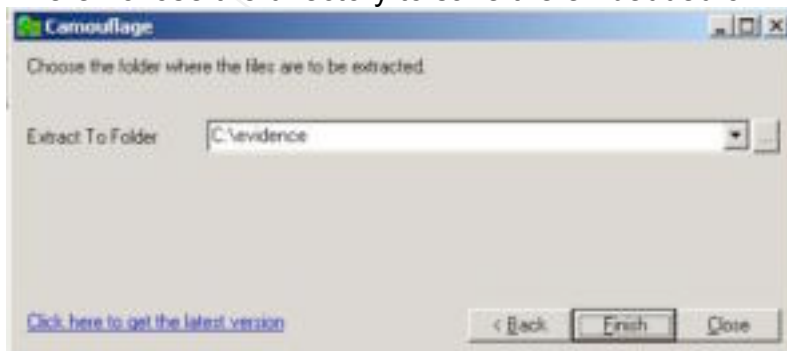


Figure 30

Selecting the “Finish” button extracts both the hidden Opportunity.txt, and the document to the selected directory. The output of Opportunity.txt follows:

“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”

Figure 31

Figure 31 reveals the intentions of Mr. Leszczynski, it also speaks of providing schematics of a “not yet available” device. The next sentence speaks of how they are available with the words First Name in quotations. “First Name” is referring to the first name of the documents that have objects embedded or “stegged” within them. This was determined both by breaking the password algorithm, and by performing the uncamouflage process.

When Uncamouflaged the Password_Policy document revealed several items.

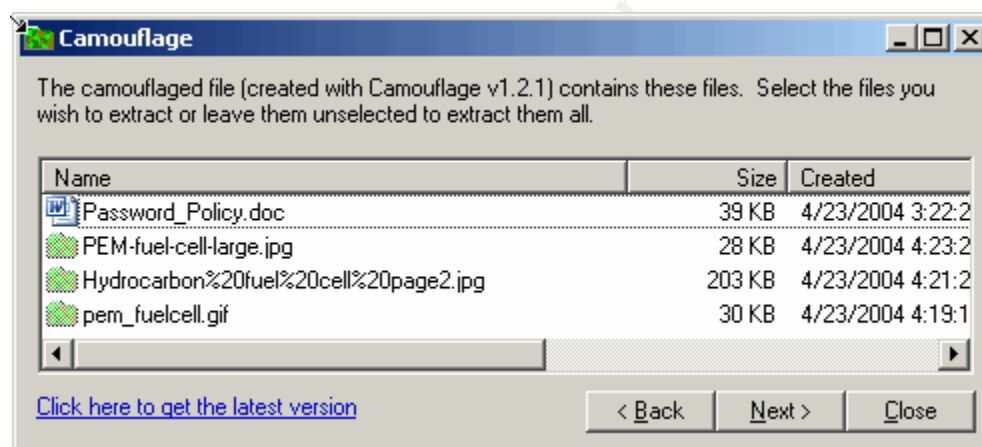


Figure 32

Figure 32 displays trade secrets embedded in a policy document. These Pictures are of a formula, and two graphical representation of a PEM fuel cell. Mr. Leszczynski was in possession of detailed diagrams of proprietary products which were hidden inside of regular policy documents. These pictures would let a competitor’s engineers duplicate our product very easily. Screenshots of the revealed documents and images follow.

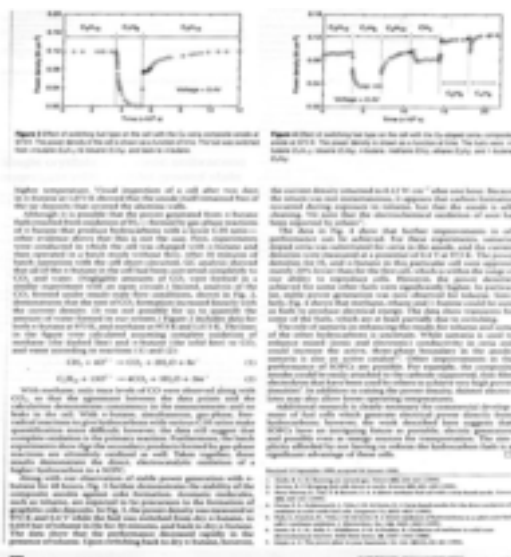


Figure 33

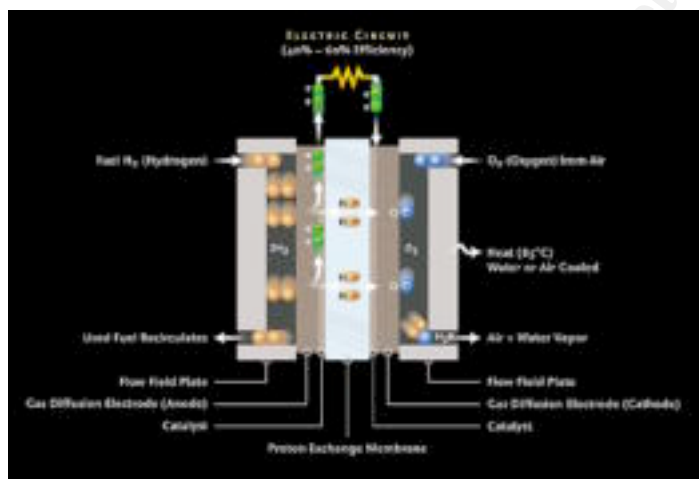


Figure 34

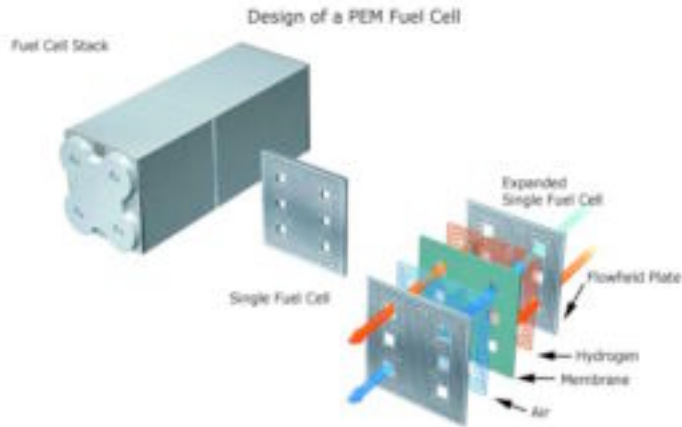


Figure 35

Using the naming scheme utilized by Mr. Leszczynski, the password “Remote” was used when uncamouflaging the Remote_Access_Policy document. A single file was revealed, the screenshot follows.

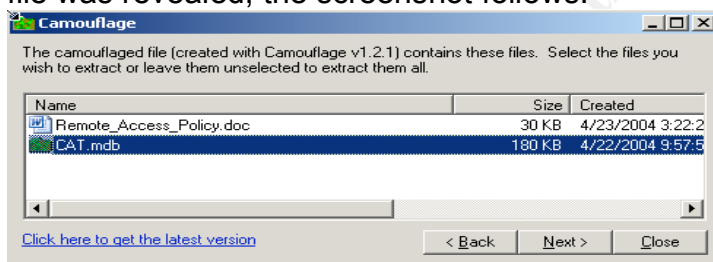


Figure 36

The database revealed in figure 36 contains a list of clients with phone numbers and accounts.

First	Last	Phone	Company	Address	Address1	City	State	Zipcode	Account	Password
Jerry	Esposito	703-233-2648	Cook Labs	345 Main St		Alexandria	VA	22231	esposito	p4RSH6M
David	Lee	410-677-7723	Double J's	11561 W 27 St		Baltimore	MD	20276	lee27M	LSWSPg
Marie	Horner	866-554-0822	Tech Vision	300 Lane Grove Lane		Wichita	KS	30789	horner	01A26a3
Lenny	Jones	800-234-4444	King Labs, Inc.	700 King Labs Ave	Suite 300	Bloom	MS	39533	horner	Yk7SP4A
Jeff	Hayes	677-Get-done	Quick Printing	99 E. Grand View Dr		Omaha	NE	56206	jones	882p48RH
Roger	Forrester	404-893-5521	Big Sky First	90 Old Saw Mill Rd		Billings	MT	59102	hayes	3R33b71
Edward	Cash	212-686-2312	TCFL	188 Greenville Rd		Austin	TX	77239	forrester	u4QW6V
Steve	Bei	212-682-0897	E & C Inc.	75 S. King St	Suite 300	Santa Barbara	CA	90124	cash	0B601C
Jodie	Kelly	616-833-0129	Island Labs	65 Koi Way		Honolulu	HA	96891	bei	JDC0a26
Patrick	Roy		Data Movers	7256 Sweenish Ave	Suite 110	Waltham	U.K.	15222 GRG	kelly	rmDEN06
			The Magic Lamp	4150 Regents Park	Box #170	Calgary	CAN	843150F	roy	uag5000

Figure 37

Figure 37 displays the most recent client list that was obtained. Mr. Leszczynski used Camouflage to hide Ballard Industries trade secrets with the intention of selling them to our competitor. Furthermore, he attempted to remove a client list to enhance the competition's ability to steal Ballard Industries active clients.

Examination Conclusion

Based on recent events, loss of sales and clients purchasing from Rift inc., it was reasonable to assume that corporate espionage had taken place. The Information Security Office has verified in this report that Mr. Leszczynski intended to sell Ballard's proprietary information. It is also reasonable to assume that Mr. Leszczynski is responsible for the events pertaining to our recent loss of customers. System Administrators throughout Ballard industries need to check for the presence of camshell.dll or Camouflage on their Windows based systems. All systems should be searched for CamShell.dll, and the ISO notified if it is found. Additional research revealed that even if the software is uninstalled, the registry keys remain. Having installed and removed the software in the forensics lab a comprehensive list of instructions is being prepared for Ballard personnel. In addition, I suggest the ISO perform forensics on any machine Mr. Leszczynski had direct access to, and possibly any machines in spaces he has accessed recently. These steps are necessary to adequately pursue the containment, eradication and recovery phases of an incident. The Incident response Procedure for Ballard industries should be invoked to provide full authority for the ISO to continue this investigation. Mr. Leszczynski's work computer and home computer should also be analyzed to determine the extent of the damage. It may be necessary for his home machine to be analyzed by members of local law enforcement since it may be privately owned.

Legal Implications

The Legal implications section of this report is not intended as legal advice. This section details the policy and procedure violations discovered during the forensic analysis process. In addition to Ballard Industries policy and procedure violations, it is possible that several laws were broken. According to the very documents Mr. Leszczynski was hiding the proprietary information in he is subject to termination and criminal/civil action. His employment can be terminated for using weak passwords on all of his stolen/stegged files. Mr. Leszczynski violated the password policy on all three files. Section 5.0 of each policy states, "Any employee found to have violated this policy may be subject to disciplinary action, up to and including termination of employment." According to the Information Sensitivity Policy, he is subject to civil and criminal proceedings based on company policy. The Acceptable Use Policy from this company states the following:

Abuse of policies or standards, abuse of IT resources, or abuse of other sites through the use of IT resources may result in termination of access, disciplinary review, expulsion, termination of employment, legal action, and/or other

appropriate disciplinary action. Notification will be made to the appropriate office (e.g., appropriate office for student conduct matters, Human Resources, General Counsel, the police department with campus jurisdiction) or local and federal law enforcement agencies.

The definition of what's acceptable is located in the same document.

(a) No one shall knowingly or willingly interfere with the security mechanisms or integrity of IT resources. Users shall not attempt to circumvent data protection schemes or exploit security loopholes.

(b) No one shall knowingly create, install, exec, or distribute any malicious code or another surreptitiously destructive program on any IT resource, regardless of the result.

In addition to violating company policy the following laws may have been broken:

The Federal Computer Fraud & Abuse Act
18 U.S.C. §1030(a)(2) & (c)(2)(B)(i)-(ii)

Criminal Copyright Laws Violations
18 U.S.C. 2319 & 17 U.S.C. 506a

Criminal Trade Secrets Violations
18 U.S.C. 1831, 1832

Ballard Industries must demonstrate loss of revenue totaling \$5,000 to report these losses as a felony. Based on the broad definition of "protected" computer in The Federal Computer Fraud and Abuse Act, we can show that he intended to defraud Ballard Industries of its proprietary design information and perhaps affect interstate commerce. For federal prosecution the current minimum dollar value needs to total \$75,000. The ISO believes that Ballard industries has potentially lost millions of dollars over the life of the products currently in use. These thresholds having been met, I would advise turning this report over to a legal representative to pursue compensation from Mr. Leszczynski, and perhaps Rift inc..

© SANS Institute retains full rights.

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

A Touch of Superiority in Linux

I have a great story to tell about some systems that are being attacked/hacked and what I am doing in conjunction with other campuses and law enforcement to catch the person who is doing it.

Synopsis

Over the past several months, the Information Security Office has been noticing an increase in compromises and we do not think it is simply because the students have returned from their summer hiatus. Perhaps the most interesting thing about this increase is that it affects only Unix/Linux platforms. Talk about a group of Sysadmins who have become overconfident and have so loved telling MS users “If you would just run a secure OS...”. I have actually been waiting for a couple of years to see something like this. It is interesting that it finally happened, but more interesting what is happening.

A group of four people from our Information Security Office was asked to attend a security briefing at a federal government facility. We walked in, exchanged hellos, and were then told of an ongoing problem where someone was compromising super-computing centers and government facilities almost at will. Worse than that was the taunting e-mail from the victims own server with a list of IP addresses he “owned”. We were then given a list of our systems that had been affected. Oh well, that is not embarrassing! As a college campus, we often fight the perception of tighter security restricting academic freedom, or the ability to perform research so we know we are a good target. The systems that were compromised were all UNIX/Linux systems, and several of them had great system administrators. Several portions of the ongoing investigation are sensitive but I believe I can provide enough forensic detail to show what is being done even though the perpetrator has been identified but not yet arrested. We went through several machines at the beginning before we were able to establish the hacker’s pattern. The following pages will detail how we have “fingerprinted” his actions and are now prepared to help track what he does when the attacks happen here.

System Details

The machine I am using was acquired from a department with several thousand students, and hundreds of employees. This machine was a departmental web, file, and mail server with 70 GB of drive space running Red Hat 7.3. In addition to serving the population while they are on campus, they are tasked with providing these services as their constituents travel. This means the machine must maintain a public IP address, as the current staff cannot adequately train its personnel to insist on secure services. Red Hat 7.3 is quite old and outdated; often system administrators will become comfortable with an operating system or service and continue to use it despite it being vulnerable. The machine had only two partitions of linux-ext3 they were / and /home. Slash is 7.5 GB, and the rest was in /home. I will attach relevant screen shots after the system description. The system is connected to our network via 100mb CAT5 with a public IP address. The system administrator has at least two other major responsibilities besides managing this aging server. Having multiple job titles is a philosophical change we are pushing for. I remember when it was perfectly acceptable to be a part time network/system administrator. There are still a lot of companies who want their “security administrator” to also be the network person, and maybe the applications person. These days, there is no way to keep up with the constant barrage of application, server upgrades and vulnerabilities. We believe this department, and hopefully others, will soon be employing someone full time in the system administrator capacity. The following diagram demonstrates how this server is connected to our network.

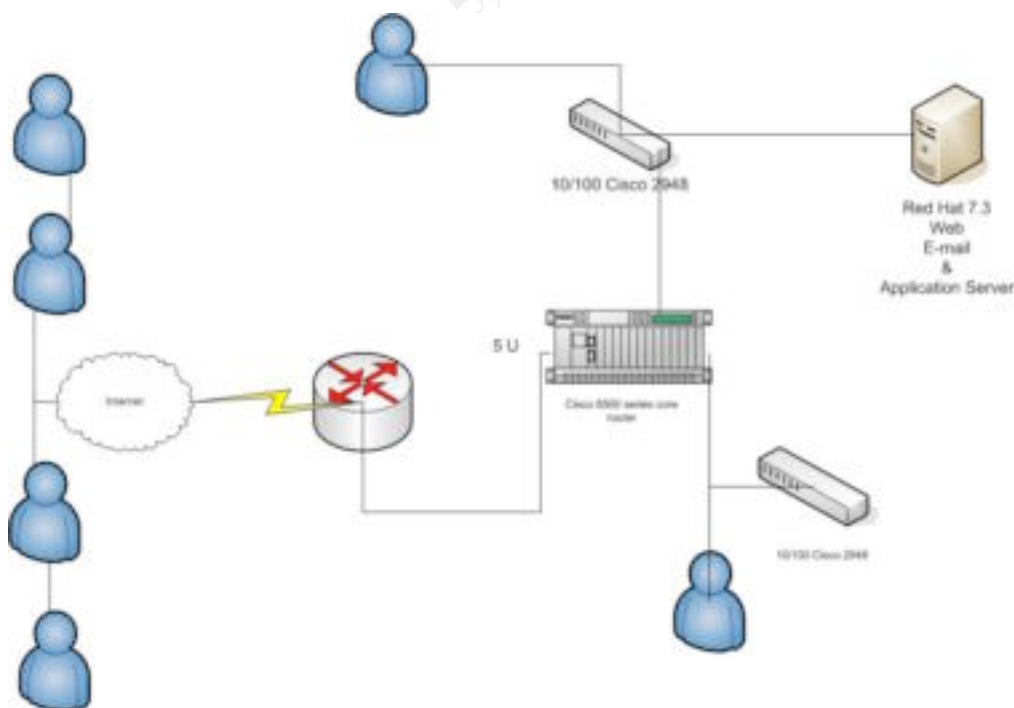


Figure 38

Hardware Seizure

The Information Security Office makes a practice of seizing the computer tower or rack mount chassis for incidents that involve servers. This is especially true if the system in question contains any confidential information such as SSN, credit card numbers, FERPA^{xiv}, HIPAA^{xv}, or GLBA^{xvi} information. The Central Information Technology department has enough other servers and drive space on campus to give administrators a safe place to maintain their data while we are in possession of their machine. This provides a more efficient solution when RAID arrays are employed, or if the server has several hard disks in it. The system was “live” when it was encountered so the disk imaging happened immediately. This step will be detailed in the section titled image details. Hardware is seized until multiple copies or disk images can be created, this done to prevent the rebuilding of machines by system administrators. Because machines are in short supply for smaller departments, sysadmins attempt to return them to production status rapidly. Below is a sample of the evidence tag collected from this system. In addition to the evidence tag, I will attach a chain of custody as Appendix C without the identifying information.

Tag #: 503116

Description: Dell Power Edge 2300 running Red Hat Linux 7.3

3 SCSI HDDS w/ 36.4G each and 1 Ethernet card

Serial number verified against asset tag

Figure 39

The tag number matches the inventory number of that particular machine. It is beneficial for our office to tie the evidence tag to the actual inventory maintained by another office. This reduces our administrative overhead and allows for some built-in redundancy. This allows the Information Security Office to have a separate group that can verify what hardware was seized against a proprietary asset tag. The proprietary asset tag inventory contains the packing slip, or a copy, which contains items such as hard drive serial numbers and MAC addresses from Ethernet cards.

The location of the victim machine, including the Ethernet port number, is noted when filling out the description portion of the chain of custody paperwork. This allows me to cross reference the system owner with the networking group who has their own database of all port numbers on campus. This is another way to make sure all of the data is accurate and double or triple checked by different groups of people. One of the other nice things about different groups owning these inventories and databases is that they can provide me with forensic evidence without having to be brought into the investigation. This helps keep only the people who need to know closest to the evidence further preventing contamination.

Image Media

I used dd to create the image of the media, I have found that the Helix CD provided in class and also found at Drew Fahey's website^{xvii} is absolutely the best tool out there. The built in ability to use Netcat in the grab function is both convenient, and forensically sound. This tool is cross-platform compatible and capable of taking a live or dead image. This machine had been removed from our network by removing the network cable. This is common practice among system administrators who learn their system has been compromised. As stated before, this image was live, so the CD was placed in the drive and the static binaries used to perform the following functions. Static binaries are important when making this case; I couldn't use anything on the compromised system since it could contaminate the evidence. This particular hacker likes to Trojan some interesting things, and I feel it is best to consider nothing on the victim machine usable as I collected my evidence. After placing the Helix CD in the drive and mounting it, I created an image, transferred the image across the network, and "fingerprinted" the image with an MD5sum. The commands were as follows:

```
mount
```

```
fdisk -l
```

These commands were used to determine how the drive partitions were structured.

```
md5sum /dev/sda6  
(wait wait wait)
```

This command created an md5sum of the / partition of the raid array.

```
md5sum /dev/sda7  
(go home take a nap)
```

This command created an md5sum of the /home partition of the raid array, these were the only two partitions on the machine. The network cable was plugged in and the interface enabled. Next I created the actual image using dd as follows:

```
dd if=/dev/sda6 | /mnt/cdrom/Static-Binaries/linux_x86/nc xxx.xxx.xxx.xxx 221
```

This command created a bit image of the drive and sent it to my forensics workstation (xxx.xxx.xxx.xxx) on tcp port 221. I used dd to ensure a complete re-creation of every bit on the drive. On the remote forensics workstation, I ran the following command to create a netcat listener.

```
nc -l -p 221 > sda6slash  
for the first partition
```

```
nc -l -p 221 > sda7home  
for the /home partition
```

The same was done for the other partition on the drive:

```
dd if=/dev/sda7 | /mnt/cdrom/Static-Binaries/linux_x86/nc xxx.xxx.xxx.xxx 221
```

After the creation/transmission of the image, I needed to make sure nothing was “altered” as it traversed our network.

```
[root@redacted]# md5sum /dev/sda6  
62d4fa0f31b58c65794aa9266a7cb11f /dev/sda6  
[root@redacted]#
```

Figure 40 md5sum for the first partition.

```
[root@redacted]# md5sum /dev/sda7  
8b8e55e903a7f4c60e58614ef8601757 /dev/sda7  
[root@redacted]#
```

Figure 41 md5sum for the second partition

```
MD5: 62D4FA0F31B58C65794AA9266A7CB11F  
Host Directory: /evidence/redacted
```

Figure 42 Autopsy displaying the md5 for /dev/sda6.

```
MD5: 8B8E55E903A7F4C60E58614EF8601757  
Host Directory: /evidence/redacted
```

Figure 43 Autopsy displaying md5 for /dev/sda7.

I checked the md5s every time I opened the case or moved the image at all. As long as the md5 doesn't change, the evidence has not been altered. As you can see, I have chosen to use Autopsy for this investigation. I am more familiar with Autopsy and I prefer to work in Linux for the speed and flexibility. Now that I had a good image on my forensic system, I wanted the victim machine off the network and stored until I am finished. This is always my recommendation to the system owner, but in the end it is a business decision and I leave it up to them whether or not to rebuild and return the machine to service. My recommendation was to re-install the operating system, and apply our operating system hardening procedures before it was given a public IP address. If we keep the machine, hardware is maintained in an access controlled room that is also under video surveillance. There is a 24x7 operations center directly across the hall to assist with the physical security of seized evidence. Paper files are maintained in a locked inventoried cabinet, with only members of the Information Security Office

possessing keys. The investigator or any member of the Information Security Office can provide a viable chain of custody for any seized evidence in addition to displaying if a copy of any paperwork was created.

Media Analysis

For analysis, I am going to use Autopsy and TSK^{xviii} developed by Brian Carrier. Autopsy leaves the image in a “clean” space. The images for Autopsy sit in their own directory logs, reports, and the actual forensic tools stay in their own directories and make no modifications to the actual dd image. This fact can be verified by using the md5sum command at any time against your images. You can, and should, check the md5 randomly to ensure your images have not been tampered with.

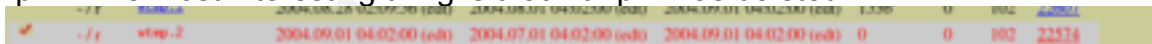
This is the most heavily deleted area in the image; from here, I extracted some key words to search and narrow my focus just a little. It was noted later in the actual investigation that this user account appeared on some machines in other departments. According to the system administrator this is a normal user with no special permissions who never causes problems. This user’s name was scattered throughout different directories and also appeared in areas that only the root, or superuser, account should have access to.

I also want to see what programs are executing on startup and what’s living in /etc. This step will help to establish how the hacker or hackers maintain access after the machine is compromised. Finding this can also assist you in locating the original compromise in many operating systems. As I looked through /etc/rc.d I saw this deleted entry.



Figure 44

Someone deleted sshd, but according to the startup processes ssh was still available when the machine started. This daemon, or service, being deleted at some point yet still available is not a good sign. Nothing else in the startup appeared abnormal. It runs httpd, sshd, and a few other normal processes. Another deleted file I found in the aforementioned user's home directory is called Ettercap^{xix}. Ettercap is a program that allows you to sniff a switched network. It does this by ARP poisoning^{xx} your switch. This lets the hacker pretend to be the local network's gateway. In essence, you become the path to the internet or network for any user connected to that device. After finding Ettercap I performed a search using the strings search. I found Ettercap in /var/tmp/Ettercap-0.6.b and once inside that directory I found a file containing several employees' usernames and passwords for a variety of different systems both on and off campus. We currently do not require the use of secure protocols, but this incident is changing that. Besides usernames and passwords, I was noticed our public read write string for all of our network devices in the same file. I cannot show a screen shot of the directory, it would require obfuscating too much data to make it worth while. What I can say is that I see the 28th of the month a lot again. This person has collected enough usernames and passwords that we could be in serious trouble. In the same directory there was another file that was titled ssh.passwords. Ninety-percent of all users on this campus use insecure protocols when authenticating to machines around campus. The other ten percent use SSH so if it has been trojaned our perpetrator will have a good collection. This is one of the reasons not to trust anything on a victim machine. If I had chosen not to respond in person, but had instead used SSH my account information would have been compromised just like everyone else. The next step was to check the /var/log directory and check wtmp. I found a critical piece of evidence when I looked in that directory. I see files named wtmp, wtmp.1 and wtmp.2. The most interesting thing is that wtmp.2 was deleted.



File Name	Size	Permissions	Owner	Group	Modified
./log wtmp.2	0	-rw-r--r--	root	root	2004.09.01 04:02:00 (UTC)

The last modified time was very near the 28th, close enough to make me really zero in on that area of the timeline. Before that though I want to read through these files and see who is logging on from where. We can do this because the metadata^{xxi} is intact. I can take that inode number and start looking around at other nearby inodes since this one has not yet been overwritten. To sum up so far, I noticed that user oleg2 was deleting a lot of information; much of the deleting appears to be files created on the 28th of the month in question. Some of the things modified/deleted were sshd and Ettercap. Ettercap was found to be intact but buried in an obscure directory. Ettercap had been collecting information, and the filename suggests it was all collected on the 28th. Now I dug into the wtmp files to find out where oleg2, and other users, like to log in from. The wtmp of a very busy machine can be extremely large, and have hundreds or even thousands of login entries. I used the Unix command grep to pull out the relevant entries but there was still a lot of information. The command grep is a search command that looks for matching text. This can be used on files,

directories, and even running processes. The user in question tended to log in from a workstation registered to him, sometimes a cable connection from his house, and once from a machine in a different department. The login from the external department occurred very near the 28th. I now wanted to focus more on this user, and the days near the 28th. From past experience with older Linux systems, I like to run strings against certain binaries. The two binaries I am going to focus on are the ssh daemon, because it was deleted once, and /sbin/init. Init is the process upon which every other process is dependant. Since the machine was not running any abnormal processes when it was starting up means we need to check closely the process that controls all other processes, /sbin/init. The init file was full of foul language and terminology, this is evidence of the suckit rootkit^{xxii}. I now needed to check the MACtimes for init and try to find out how and when suckit was installed.

I did this by taking a closer look at the timeline, the text file generated by Autopsy was over 100 megabytes. This is an abnormally large timeline, which is evidence of the machine having been in service for a long time. I will show the most relevant pieces I found, and I will attach a good bit of what it contains. The operating system appears to have been installed quite awhile ago, and was not patched or updated near as often as it should have been. As stated previously this machine was in high demand by this department. It was used everyday by many people, and provided many services.

Timeline Analysis

The timeline for this investigation was created as the image was being imported into Autopsy. The timeline was generated using the following steps in the Autopsy Forensic Browser. During the case creation Autopsy is instructing the Sleuthkit to run commands. The same commands that recover deleted information from available inodes, ils and fls, also output their findings in MACtime format when the -m switch is used. By default Autopsy instructs Sleuthkit to use the -m switch which outputs the data in MACtime format. The commands ils and fls search allocate, unallocated, and unallocated metadata in the image's inodes. The output is a "body" file, then since the output was generated in MACtime format, I tell Autopsy to generate an actual readable timeline. This allowed me to view the Modified, Accessed and Changed times of files and directories. Below are the most relevant excerpts from the timeline.

```
Sat Aug 28 2004 02:09:20
10080 m.. -/rwxr-xr-x root/w  root 866429 /usr/lib/apache/mod_rootme.soxxiii
```

The mod_rootme alerted me to a possible vulnerability in the version of the Apache web server. Four seconds later I noticed another entry that seemed suspicious

```
Sat Aug 28 2004 02:09:24
```

```
0 .a. -/-rw-r--r-- root/w root 892930 /etc/httpd/conf/httpd.conf~ (deleted).
```

Httpd.conf is the configuration file for the web server. Altering, or in this case deleting, this file can grant you additional permissions and control over this service. The speed with which the httpd.conf file was deleted after the exploit indicated that I was seeing a scripted action.

I used Google to perform some research on the file mod_rootme.so. A well known hacker site provided the following description:

mod_rootme is a very cool module for the Apache 1.3 series that sets up a backdoor inside of Apache where a simple GET request will allow a remote administrator the ability to grab a root shell on the system without any logging.

This would indicate that the compromised system had been running a vulnerable version of Apache. Navigating to the directory where Apache was installed revealed that it was indeed running the 1.3 series which is vulnerable to this exploit. A few seconds after those entries user oleg2 accessed and changed his .bash history. This is where the operating system was storing the commands executed by this user.

```
97 .ac -/-rw----- oleg2 users 1591311 /home/oleg2/.bash_history
42116 .ac -/-rw-r--r-- root/w root 75829 /usr/lib/cracklib_dict.pwi
828363 .ac -/-rw-r--r-- root/w root 75828 /usr/lib/cracklib_dict.pwd
1024 .ac -/-rw-r--r-- root/w root 75967 /usr/lib/cracklib_dict.hwm
```

These files were also created in rapid succession. The hacker had root and was loading dictionaries in order to crack user's passwords should it become necessary to retain access in the future.

```
Sat Aug 28 2004 02:18:54 689038 m.c -/-rw-r--r-- root/w root 12292 /var/tmp/ettercap-0.6.b.tar.gz
Sat Aug 28 2004 02:18:57
3893 .ac -/-rw-r--r-- 1000 users 727159 /var/tmp/ettercap-0.6.b/src/include/ec_gtk.h
1874 ..c -/-rw-r--r-- 1000 users 256141 /var/tmp/ettercap-0.6.b/src/ec_dissector_nntp.c
365 ..c -/-rw-r--r-- 1000 users 1880190 /var/tmp/ettercap-0.6.b/plugins/imp/Makefile.in
529 ..c -/-rw-r--r-- 1000 users 1880181 /var/tmp/ettercap-0.6.b/plugins/Makefile.in
6722 .ac -/-rw-r--r-- 1000 users 596097 /var/tmp/ettercap-0.6.b/src/missing/include/if_arp.h
6438 ..c -/-rw-r--r-- 1000 users 256128 /var/tmp/ettercap-0.6.b/src/ec_dissector_bgp.c
```

The above excerpt shows Ettercap being loaded and extracted onto the computer about one minute later. The timeline was more than 500 pages long, and I was unable to ascertain when the operating system was installed. For this investigation that was not very useful. There was no evidence of any significant operating system or major updates until one day after the system administrator

was contacted by the Information Security Office. The Sysadmin then made a major update to the system. Timeline output verification follows:

```
Wed Sep 08 2004 07:17:40 17 .a. -/lrwxrwxrwx root/w root 616734 /usr/lib/kde2-compat/lib/libkdesu.so.1 ->
libkdesu.so.1.0.0
15 .a. -/lrwxrwxrwx root/w root 32868 /lib/libanl.so.1 -> libanl-2.2.5.so
19 .a. -/lrwxrwxrwx root/w root 393431 /usr/lib/kde1-compat/lib/libkfm.so.2 -> libkfm.so.2.0.0
23 .a. -/lrwxrwxrwx root/w root 616768 /usr/lib/kde2-compat/lib/libkmedia2_idl.so.0 ->
libkmedia2_idl.so.0.0.0
13 .a. -/lrwxrwxrwx root/w root 32873 /lib/libm.so.6 -> libm-2.2.5.so
19 .a. -/lrwxrwxrwx root/w root 32888 /lib/libthread_db.so.1 -> libthread_db-1.0.so
24 .a. -/lrwxrwxrwx root/w root 616700 /usr/lib/kde2-compat/lib/libartsflow_idl.so.0 ->
libartsflow_idl.so.0.0.0
16 .a. -/lrwxrwxrwx root/w root 616676 /usr/lib/kde2-compat/lib/libDCOP.so.1 ->
libDCOP.so.1.0.0
31 .a. -/lrwxrwxrwx root/w root 616730 /usr/lib/kde2-compat/lib/libkdeprint_management.so.0
-> libkdeprint_management.so.0.0.1
22 .a. -/lrwxrwxrwx root/w root 32828 /lib/libnss_compat.so.2 -> libnss_compat-2.2.5.so
15 .a. -/lrwxrwxrwx root/w root 616752 /usr/lib/kde2-compat/lib/libkio.so.3 -> libkio.so.3.0.0
19 .a. -/lrwxrwxrwx root/w root 616804 /usr/lib/kde2-compat/lib/libmcp_mt.so.0 ->
libmcp_mt.so.0.0.0
17 .a. -/lrwxrwxrwx root/w root 393429 /usr/lib/kde1-compat/lib/libkfile.so.2 -> libkfile.so.2.0.0
17 .a. -/lrwxrwxrwx root/w root 32885 /lib/libpthread.so.0 -> libpthread-0.9.so
19 .a. -/lrwxrwxrwx root/w root 393421 /usr/lib/kde1-compat/lib/libjscrip.so.2 ->
libjscrip.so.2.0.0
```

This was the first of several screens of data that detail the system undergoing the only major update I could find in the timeline. This happened after the sysadmin was instructed to not change anything on the machine.

Recover Deleted Files

The first thing I like to do with Autopsy is recover the deleted files in the file analysis window. The deleted files are actually recovered when the case is being created, and the image I created in the previous section is being imported into the program. The deleted files are recovered when the Autopsy Forensic Browser instructs Sleuthkit to run commands “under the hood.” The commands that are most critical when importing the image are `ils` and `fls`. The command `fls` searches inode, or disk clusters and displays the data that was stored there, and the command `ils` stands for Inode Lister it will be used to look for possible deleted data, and recover it. These two commands are how I recovered deleted data, providing that the deleted area has not yet been overwritten with new data. As I stated previously this image is very large, as I examined the deleted files I noticed a particular home area that is missing a lot of data. Because of the size of the image and the fact that it was so highly used it was necessary to try and zero in on certain deleted areas. I chose to look for dense concentrations of deleted data, either based on the date deleted or the location the files were deleted from.

~/home/oleg2/.X11/unixtool.conf	2000.05.10 (99.33.40 (oct))	2000.05.10 (99.33.36 (oct))	2
~/home/oleg2/.X11/unixtool.conf	1999.07.28 (17.49.51 (oct))	2000.11.19 (04.29.39 (oct))	1
~/home/oleg2/.X11/unixtool.conf	1999.08.17 (14.47.44 (oct))	2000.11.19 (04.17.21 (oct))	1
~/home/oleg2/.X11/unixtool.conf	2002.02.27 (18.10.20 (oct))	2004.09.02 (04.21.46 (oct))	2
~/home/oleg2/.X11/unixtool.conf	2002.04.18 (17.35.54 (oct))	2004.09.08 (07.33.42 (oct))	2
~/home/oleg2/.X11/unixtool.conf	2000.10.09 (12.46.29 (oct))	2000.10.09 (12.46.24 (oct))	2
~/home/oleg2/.X11/unixtool.conf	2000.10.09 (12.46.24 (oct))	2000.10.09 (12.46.24 (oct))	2
~/home/oleg2/.X11/unixtool.conf	1996.04.12 (03.55.56 (oct))	2002.08.13 (21.43.36 (oct))	1
~/home/oleg2/.X11/unixtool.conf	2000.10.09 (12.46.24 (oct))	2000.10.09 (12.46.24 (oct))	2

Figure 45

Several files appear to have been deleted on the 28th of the month and all most of the deletion is in this user's home directory. Autopsy can either show the investigator all of the deleted files on one page, or they can be displayed where they were last residing. Figure 44 is a display of deleted files that have been recovered, and are being displayed where they last resided. The image and amount of deleted files made it nearly impossible to use the function Autopsy provides of sorting all deleted files to one screen. This was inconvenient, but ended up having little effect on the overall analysis. Having used Autopsy to sort Sleuthkit's recovery of the deleted files I proceeded to the analysis of the actual media. At this point the recovery work was done, but the analysis needed to be included with other portions to make them relevant.

Searches

Performing searches on such a large image allowed me to parse through an enormous amount of data quite rapidly. The timeline and media analysis portions gave me several terms and phrases to search for. I used the keywords, rootme and cracklib to perform strings and grep searches throughout the image. The Autopsy Forensic Browser provides a user-friendly interface in which to run strings and grep searches. The ability to run these searches was especially beneficial during the media analysis section when I discovered the presence of Ettercap. It was again beneficial after finding the apache exploit I could then grep search throughout the entire image for other occurrences of those exploits. This particular hacker likes to collect as many usernames and passwords as possible so that he can exploit as many machines and accounts as possible. This allows you to retain access in case you exploit is discovered and cleaned up. That is the only reason to run crack programs against /etc/passwd and /etc/shadow when you are already root. I decided to run john^{xxiv} against the unshadowed /etc/passwd and /etc/shadow files. I found a lot of default system passwords, and many plain dictionary words. To see exactly what has been trojaned I want to run chkrootkit, and get a comprehensive listing. Based on other information collected I also want to take a look at that other system on campus where I saw oleg2 log in from. I can also now use the terms Ettercap, oleg2, and the script names I found to conduct keyword searches on the images I

took. The oleg2 account was targeted because that user was very busy. The accessing of Oleg2' bash history in the timeline analysis portion must have shown the hacker that this person was on the system often, and was quite busy while logged in. This makes that account and that home area an excellent place to store files and a good account to run commands from. The bash history can only hold a certain amount of data and is constantly being overwritten as new data is being added.

The results of some of the searches performed are discussed in the media analysis section of this paper.

Conclusion

When the Information Security Office went to investigate that machine prior to this machine being finished we couldn't believe what we found. Linux/UNIX machines all over the campus are hacked in a similar manner. This hacker has some interesting techniques that allow us to "fingerprint" his/her actions. They have Ettercap hidden with a trojaned sshd running. Almost all of them have the suckit rootkit as evidenced in /sbin/init. It has been determined by the consortium I am participating with that this person is using our campus as "throw away" hosts. We are a hop, and a good one, for him/them to acquire higher profile targets. The fact that we are a throw away host is evidenced by the fact that usernames and passwords are collected, yet data is never altered or destroyed. This allows him to have thousands of legitimate usernames and passwords to hundreds of different machines. Even if we found all of the machines and rebuilt the entire OS he could still SSH into any one of them. We would have to eradicate every trace and simultaneously have 25,000 people change their passwords. The hacker leaves other interesting clues, such as sometimes wtmp files are deleted, and other times they are not. This would tend to point towards there possibly being more than one hacker. So far this person or group of people has used the exact same tools on every machine. Things are not always in the same place, but some things are always the same. The hacker is very agile once he is on a system, he/she gives all scripts and tools a one character name allowing for great speed in compromising a system or network. A colleague from another compromised campus related that the hacker once started a talk session with him and was actively hacking the system as he taunted the sysadmin. This was particularly impressive since the sysadmin claimed he could touch type about 80 words per minute and he could not keep up with this hacker. Another colleague from a different location watched as the hacker entered a new system that was set as a trap and had it successfully compromised within 15 seconds and had moved on to the next target. The honey pot machine was also running an innocuously named script that was almost immediately discovered and terminated by the hacker. The script was designed to transfer all keystrokes to a remote location. The hacker or group of hackers has another identifying trademark; he/they seem completely unaware of some other aspects of computing and networking. For example he seems to have no knowledge of the

concept of syslog and syslogd, or the ability to passively sniff a network from a span port. Those two factors are what have allowed us to gain the most ground on an otherwise very impressive perpetrator. Stolen accounts and the presence Ettercap are also viable fingerprints. The amount of stolen accounts makes this hacker very difficult to track. With thousands of possibly compromised accounts the hacker or hackers no longer have to use an exploit to gain access to systems on our campus. They can make legitimate logins to several systems. To mitigate this particular problem, a plan is in the works to have more than 25,000 users change their passwords. These new passwords will also have to meet strength checking requirements. In addition to all new passwords, insecure protocols will soon be removed from the network. These are two major steps towards a good defense in depth policy that should help to secure the information technology resources on this campus.

In all more than 15 machines at this location have been successfully compromised by this hacker or group of hackers. Every new compromised machine is now considered part of this case unless it can be ruled out. We have been working closely with law enforcement and they now know whom this person is and expect an arrest within weeks

© SANS Institute 2005, Author retains full rights.

Appendices

Appendix A

Lame Camouflage Password Scheme

A 43	A D4	A 3B	A 63	A 4D	A E7	A 55	A A0
B 40	B D7	B 38	B 60	B 4E	B EF	B 56	B A3
C 41	C D6	C 39	C 61	C 4F	C E5	C 57	C A2
D 46	D D1	D 3E	D 66	D 48	D E2	D 50	D A5
E 47	E D0	E 3F	E 67	E 49	E E3	E 51	E A4
F 44	F D3	F 3C	F 64	F 4A	F EO	F 52	F A7
G 45	G D2	G 3D	G 65	G 4B	G E1	G 53	G A6
H 4A	H DD	H 32	H 6A	H 44	H EE	H 5C	H A9
I 4B	I DC	I 33	I 6B	I 45	I EF	I 5D	I A8
J 48	J DF	J 30	J 68	J 46	J EC	J 5E	J AB
K 49	K DE	K 31	K 69	K 47	K ED	K 5F	K AA
L 4E	L D9	L 36	L 6E	L 40	L EA	L 58	L AD
M 4F	M D8	M 37	M 6F	M 41	M EB	M 59	M AC
N 4C	N DB	N 34	N 6C	N 42	N E8	N 5A	N AF

O 4D	O DA	O 35	O 6D	O 43	O E9	O 5B	O AE
P 52	P C5	P 2A	P 72	P 5C	P F6	P 44	P B1
Q 53	Q C4	Q 2B	Q 73	Q 5D	Q F7	Q 45	Q B0
R 50	R C7	R 28	R 70	R 5E	R F4	R 46	R B3
S 51	S C6	S 29	S 71	S 5F	S F5	S 47	S B2
T 56	T C1	T 2E	T 76	T 58	T F2	T 40	T B5
U 57	U C0	U 2F	U 77	U 59	U F3	U 41	U B4
V 54	V C3	V 2C	V 74	V 5A	V F0	V 42	V B7
W 55	W C2	W 2D	W 75	W 5B	W F1	W 43	W B6
X 5A	X CD	X 22	X 7A	X 54	X FE	X 4C	X B9
Y 5B	Y CC	Y 23	Y 7B	Y 55	Y FF	Y 4D	Y B8
Z 58	Z CF	Z 20	Z 78	Z 56	Z FC	Z 4E	Z BB
a 63	a F4	a 1B	a 43	a 6D	a C7	a 75	a 80
b	b	b	b	b	b	b	b
c	c	c	c	c	c	c	c
d	d	d	d	d	d	d	d

66	F1	1E	46	68	C2	70	85
e 67	e F0	e 1F	e 47	e 69	e C3	e 71	e 84
f	f	f	f	f	f	f	f
g	g	g	g	g	g	g	g
h	h	h	h	h	h	h	h
i	i	i	i	i	i	i	i
j	j	j	j	j	j	j	j
k	k	k	k	k	k	k	k
l	l	l	l	l	l	l	l
m 6F	m F8	m 17	m 4F	m 61	m CD	m 79	m 8C
n	n	n	n	n	n	n	n
o 6D	o FA	o 15	o 4D	o 63	o C9	o 7B	o 8E
p	p	p	p	p	p	p	p
q	q	q	q	q	q	q	q
r 70	r e7	r 08	r 5O	r 7E	r D4	r 66	r 93
s	s	s	s	s	s	s	s

71 E6 09 51 7F D5 67 92

t t t t t t t t
76 E1 OE 56 78 D2 60 95

u u u u u u u u

v v v v v v v v

w w w w w w w w
75 E2 0D 55 7B D1 63 96

x x x x x x x x

y y y y y y y y

z z z z z z z z

0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1

2 2 2 2 2 2 2 2

3 3 3 3 3 3 3 3

4 4 4 4 4 4 4 4

5 5 5 5 5 5 5 5

6 6 6 6 6 6 6 6

7 7 7 7 7 7 7 7

8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

.
---	---	---	---	---	---	---	---

!	!	!	!	!	!	!	!
---	---	---	---	---	---	---	---

?	?	?	?	?	?	?	?
---	---	---	---	---	---	---	---

1	2	3	4	5	6	7	8
P	a	s	s	w	o	r	d
52	F4	09	51	7B	C9	66	85 -3.jpg

1	2	3	4	5	6
R	e	m	o	t	e
50	F0	17	4D	78	C3 –Access database with list of clients

© SANS Institute 2005, Author retains full rights.

Appendix B

Timeline excerpts

```
91181 m.c -/-rw-r--r-- root/w root 12460 /var/tmp/taperaSXULm
(deleted-realloc)
31 m.c -/-rw-r--r-- root/w root 12457 /lib/modules/2.4.18-
3smp/modules.generic_string
147 m.c -/-rw-r--r-- root/w root 12462 /lib/modules/2.4.18-
3smp/modules.ieee1394map
8257 m.c -/-rw-r--r-- root/w root 12459 /var/tmp/taperIQ9NJH (deleted-
realloc)
31 m.c -/-rw-r--r-- root/w root 12457 /var/tmp/taper11DvVK (deleted-
realloc)
4920 .a. -/-rw-r--r-- root/w root 1022124 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_red.o
15556 .a. -/-rw-r--r-- root/w root 211109 /lib/modules/2.4.18-
3smp/kernel/net/wanrouter/wanrouter.o
5244 .a. -/-rw-r--r-- root/w root 1022126 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_tbf.o
6988 .a. -/-rw-r--r-- root/w root 1022121 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_gred.o
7144 .a. -/-rw-r--r-- root/w root 1022127 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_teql.o
93938 m.c -/-rw-r--r-- root/w root 12456 /lib/modules/2.4.18-
3smp/modules.dep
6228 .a. -/-rw-r--r-- root/w root 1022112 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_fw.o
8544 .a. -/-rw-r--r-- root/w root 1022117 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_u32.o
4092 .a. -/-rw-r--r-- root/w root 1022122 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_ingress.o
93938 m.c -/-rw-r--r-- root/w root 12456 /var/tmp/taperGbGYOZ
(deleted-realloc)
29 m.c -/-rw-r--r-- root/w root 12461 /var/tmp/taperTaybgc (deleted-
realloc)
166259 .a. -/-rw-r--r-- root/w root 1050772 /lib/modules/2.4.18-
3smp/kernel/net/tux/tux.o
29 m.c -/-rw-r--r-- root/w root 12461 /lib/modules/2.4.18-
3smp/modules.parportmap
60957 m.c -/-rw-r--r-- root/w root 12458 /var/tmp/taperaS8d0v (deleted-
realloc)
99566 .a. -/-rw-r--r-- root/w root 266324 /lib/modules/2.4.18-
3smp/kernel/net/sunrpc/sunrpc.o
60401 .a. -/-rw-r--r-- root/w root 1020025 /lib/modules/2.4.18-
3smp/kernel/net/rose/rose.o
```

```

8257 m.c -/-rw-r--r-- root/w root 12459 /lib/modules/2.4.18-
3smp/modules.isapnpmap
18296 .a. -/-rw-r--r-- root/w root 1022118 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_cbq.o
7584 .a. -/-rw-r--r-- root/w root 1022116 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_tcindex.o
24 m.c -/-rw-r--r-- root/w root 12463 /var/tmp/taperbHkObR (deleted-
realloc)
6752 .a. -/-rw-r--r-- root/w root 1022119 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_csz.o
147 m.c -/-rw-r--r-- root/w root 12462 /var/tmp/taperQGRnJ1 (deleted-
realloc)
6168 .a. -/-rw-r--r-- root/w root 1022125 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_sfq.o
24 m.c -/-rw-r--r-- root/w root 12463 /lib/modules/2.4.18-
3smp/modules.pnpbiosmap
91181 m.c -/-rw-r--r-- root/w root 12460 /lib/modules/2.4.18-
3smp/modules.usbmap
60957 m.c -/-rw-r--r-- root/w root 12458 /lib/modules/2.4.18-
3smp/modules.pcimap
8392 .a. -/-rw-r--r-- root/w root 1022115 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_rsvp6.o
5796 .a. -/-rw-r--r-- root/w root 1022123 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_prio.o
7192 .a. -/-rw-r--r-- root/w root 1022120 /lib/modules/2.4.18-
3smp/kernel/net/sched/sch_dsmark.o
8588 .a. -/-rw-r--r-- root/w root 1022113 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_route.o
8196 .a. -/-rw-r--r-- root/w root 1022114 /lib/modules/2.4.18-
3smp/kernel/net/sched/cls_rsvp.o

Tue Mar 23 2004 15:24:49 46952 .a. -/-rwxr-xr-x root/w root 63009
/usr/bin/sftp
Tue Mar 23 2004 15:26:14 1970 .a. -/-rw-r--r-- oleg2 users 1914883
/home/oleg2/Build/Makefile.bck
12288 .a. -/-rw----- oleg2 users 1914884
/home/oleg2/Build/.Makefile.swp
Tue Mar 23 2004 15:28:11 12288 .a. -/-rw----- oleg2 users 1916948
/home/oleg2/Build/Source/.ds.swp
Tue Mar 23 2004 15:28:13 20480 .a. -/-rw----- oleg2 users 1916975
/home/oleg2/Build/Source/.sigeom.F.swe
16384 .a. -/-rw----- oleg2 users 1916977
/home/oleg2/Build/Source/.sigeom.F.swc
24576 .a. -/-rw----- oleg2 users 1916974
/home/oleg2/Build/Source/.sigeom.F.swf

```

```

28672 .a. -/-rw----- oleg2  users  1916978
/home/oleg2/Build/Source/.sigeom.F.swb
3536 .a. -/-rw-r--r-- oleg2  users  1916979
/home/oleg2/Build/Source/china_ggckov.F_china
20480 .a. -/-rw----- oleg2  users  1916976
/home/oleg2/Build/Source/.sigeom.F.swd
Tue Mar 23 2004 15:28:14 51299 .a. -/-rw-r--r-- oleg2  users  1916989
/home/oleg2/Build/Source/pmtnew.F~
24576 .a. -/-rw----- oleg2  users  1916982
/home/oleg2/Build/Source/.sigeom.F.swa
48892 .a. -/-rw-r--r-- oleg2  users  1916987
/home/oleg2/Build/Source/pmt20new.F~
12288 .a. -/-rw----- oleg2  users  1916990
/home/oleg2/Build/Source/.singer.F.swp
16384 .a. -/-rw----- oleg2  users  1916984
/home/oleg2/Build/Source/.gustep.F.swp
Tue Mar 23 2004 15:28:15 14587 .a. -/-rw-r--r-- oleg2  users  1916993
/home/oleg2/Build/Source/ed.hup
20480 .a. -/-rw----- oleg2  users  1916994
/home/oleg2/Build/Source/.sigeom.F.swp
24576 .a. -/-rw----- oleg2  users  1916997
/home/oleg2/Build/Source/.sigeom.F.swm
20480 .a. -/-rw----- oleg2  users  1916999
/home/oleg2/Build/Source/.sigeom.F.swl
20480 .a. -/-rw----- oleg2  users  1917001
/home/oleg2/Build/Source/.sigeom.F.swj
24576 .a. -/-rw----- oleg2  users  1917002
/home/oleg2/Build/Source/.sigeom.F.swi
16384 .a. -/-rw----- oleg2  users  1917000
/home/oleg2/Build/Source/.sigeom.F.swk
24576 .a. -/-rw----- oleg2  users  1916995
/home/oleg2/Build/Source/.sigeom.F.swo
20480 .a. -/-rw----- oleg2  users  1916996
/home/oleg2/Build/Source/.sigeom.F.swn
24576 .a. -/-rw----- oleg2  users  1917003
/home/oleg2/Build/Source/.sigeom.F.swh
17 .a. -/lrwxrwxrwx root/w  root  743463  /etc/rc.d/rc6.d/K20rusersd -
> ../init.d/rusersd
14 .a. -/lrwxrwxrwx root/w  root  743669  /etc/rc.d/rc6.d/K25sshd -
> ../init.d/sshd
18 .a. -/lrwxrwxrwx root/w  root  743430  /etc/rc.d/rc6.d/K05keytable -
> ../init.d/keytable
18 .a. -/lrwxrwxrwx root/w  root  743466  /etc/rc.d/rc6.d/K30sendmail -
> ../init.d/sendmail
16 .a. -/lrwxrwxrwx root/w  root  743464  /etc/rc.d/rc6.d/K20rstatd -
> ../init.d/rstatd

```

```

20 .a. -/lrwxrwxrwx root/w root 743620 /etc/rc.d/rc6.d/K44rawdevices -
> ../init.d/rawdevices
19 .a. -/lrwxrwxrwx root/w root 743453 /etc/rc.d/rc6.d/K00linuxconf -
> ../init.d/linuxconf
13 .a. -/lrwxrwxrwx root/w root 743452 /etc/rc.d/rc6.d/K35smb -
> ../init.d/smb
20 .a. -/lrwxrwxrwx root/w root 743629 /etc/rc.d/rc6.d/K15postgresql -
> ../init.d/postgresql
15 .a. -/lrwxrwxrwx root/w root 743426 /etc/rc.d/rc6.d/K15httpd -
> ../init.d/httpd
19 .a. -/lrwxrwxrwx root/w root 743625 /etc/rc.d/rc6.d/K50snmptrapd -
> ../init.d/snmptrapd
14952 .a. -/-rwxr-xr-x root/w root 45153 /sbin/shutdown
15 .a. -/lrwxrwxrwx root/w root 743429 /etc/rc.d/rc6.d/K45named -
> ../init.d/named
16 .a. -/lrwxrwxrwx root/w root 743621 /etc/rc.d/rc6.d/K50xinetd -
> ../init.d/xinetd
13 .a. -/lrwxrwxrwx root/w root 743482 /etc/rc.d/rc6.d/K10xfs -
> ../init.d/xfs
15 .a. -/lrwxrwxrwx root/w root 743465 /etc/rc.d/rc6.d/K20rwhod -
> ../init.d/rwhod
15 .a. -/lrwxrwxrwx root/w root 743480 /etc/rc.d/rc6.d/K50snmpd -
> ../init.d/snmpd
13 .a. -/lrwxrwxrwx root/w root 743431 /etc/rc.d/rc6.d/K15gpm -
> ../init.d/gpm
13 .a. -/lrwxrwxrwx root/w root 743451 /etc/rc.d/rc6.d/K20nfs -
> ../init.d/nfs
15 .a. -/lrwxrwxrwx root/w root 743461 /etc/rc.d/rc6.d/K15sound -
> ../init.d/sound

```

© SANS Institute 2005

Appendix C

Chain of Custody

CHAIN OF CUSTODY

Tag #: _____

Description:

Name of Delivering: _____

Signature and Date: _____

Name of Receiving: _____

Signature and Date: _____

Comments:

Name of Delivering: _____

Signature and Date: _____

Name of Receiving: _____

Signature and Date: _____

Comments:

Appendix D

Evidence tag example

Tag #:

Description:

© SANS Institute 2005, Author retains full rights.

References

-
- ⁱ Web definition of write protect. URL:
http://www.webopedia.com/TERM/w/write_protect.html
- ⁱⁱ Unix command definition: mkdosfs, URL:
<http://www.linuxvalley.it/encyclopedia/ldp/manpage/man8/mkfs.msdos.8.php> (27 February 1997)
- ⁱⁱⁱ Unix Command definition: mount, URL:
<http://www.linuxvalley.it/encyclopedia/ldp/manpage/man8/mount.8.php> (14 September 1997)
- ^{iv} Unix Command definition: dd, URL:
[http://encyclopedia.thefreedictionary.com/Dd%20\(Unix\)](http://encyclopedia.thefreedictionary.com/Dd%20(Unix))
- ^v Web definition: md5, URL:
<http://isp.webopedia.com/TERM/M/md5.html>
- ^{vi} Software reference: The SleuthKit and Autopsy by Brian Carried. URL:
<http://www.sleuthkit.org/>
- ^{vii} Software reference: The Forensic ToolKit by Accessdata, URL:
http://www.accessdata.com/Product04_Overview.htm?ProductNum=04
- ^{viii} web definition: dll, URL:
<http://www.pace.ch/cours/glossary.htm>
- ^{ix} Search Engine; URL:
<http://www.google.com>
- ^x Specific dll definition: msvbvm60.dll, URL:
<http://www.liutilities.com/products/wintasksp/dlllibrary/msvbvm60/>
- ^{xi} Google search for definition: steganography, URL:
<http://www.google.com/search?hl=en&lr=&oi=defmore&q=define:Steganography>
- ^{xii} Software reference: Helix by Drew Fahey e-fense inc, URL:
<http://www.e-fense.com/helix/>
- ^{xiii} Web reference: steganography and Camouflage by Guillermito URL:
<http://www.guillermito2.net/stegano/camouflage/> (, September 16th 2002)
- ^{xiv} Web definition: FERPA, URL:
http://www.chadd.org/webpage.cfm?cat_id=23
- ^{xv} Web definition: HIPAA, URL:
<http://www.iid.state.ia.us/division/consumer/terms/default.asp#H>
- ^{xvi} Web definition: Gramm-Leach-Bliley Act, URL:
http://www.wordiq.com/definition/Gramm-Leach-Bliley_Act
- ^{xvii} Software reference: Helix by Drew Fahey e-fense inc., URL:
<http://www.e-fense.com/>

^{xviii} Software reference: The SleuthKit and Autopsy by Brian Carrier. URL:
<http://www.sleuthkit.org>

^{xix} Software reference: Ettercap by Alberto Ornaghi and Marco Valleri, URL:
<http://ettercap.sourceforge.net/>

^{xx} Web definition: ARP spoofing/poisoning, URL:
http://www.webopedia.com/TERM/A/ARP_spoofing.html

^{xxi} Web definition: Metadata, URL:
<http://www.free-definition.com/Metadata.html>

^{xxii} The Honeynet Project – Scan of the Month #29 URL:
http://www.sleuthkit.org/case/sotm_29/output/sk.txt

^{xxiii} Exploit definition: Apache mod_rootme, URL:
http://www.networkscanning.com/Apache-mod-rootme-Backdoor-VSS_13644.html

^{xxiv} Web definition: Password cracker John the Ripper, URL:
<http://www.openwall.com/john/>

© SANS Institute 2005, Author retains full rights.