



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>

Mike Aylor

GCFA v1.5

Date Submitted: Feb. 11, 2005

Forensic Analysis of Camouflage and Validation of X-Ways Forensics Tool

© SANS Institute 2005, Author retains full rights.

Table of Contents

<u>Abstract</u>	3
<u>Part I – Analyze an Unknown Image</u>	3
<u>Tools</u>	4
<u>Strings Analysis</u>	8
<u>Timeline Analysis</u>	9
<u>File Analysis</u>	11
<u>File Summary</u>	26
<u>Legal Implications</u>	28
<u>Part II – Perform Forensic Tool Validation.</u>	31
<u>Test Environment</u>	31
<u>Testing Procedures</u>	32
<u>Hypothesis</u>	32
<u>Step 1. Baselineing</u>	32
<u>Step 2. Cloning from the original image file to the X-Ways forensics image file.</u>	33
<u>Step 3. Cloning from the disk media to the X-Ways forensics image file.</u>	33
<u>Step 4. Cloning from the original image file to disk media.</u>	34
<u>Step 5. Confirm the hash calculation function of X-Ways Forensics.</u>	34
<u>Nomenclature/Terminology</u>	34
<u>Test 1 - Baselineing</u>	34
<u>Test 2 - Cloning from the original image file to the X-Ways forensics image file.</u>	36
<u>Test 3 - Cloning from the disk media to the X-Ways forensics image file.</u>	36
<u>Test 4 - Cloning from the original image file to disk media.</u>	43
<u>Test 5. Hash functions of X-Ways Forensics</u>	46
<u>Additional Analysis</u>	47
<u>Presentation</u>	47
<u>Conclusion</u>	49
<u>List of References</u>	50

Abstract

There are two parts to this practical assignment. The first part corresponds with the analysis of an unknown image, and is detailed in the synopsis of the GCFA v1.5 assignment. I was tasked to examine a floppy disk image seized from an employee who had been attempting to leave the corporate premises with it, and determine if company confidential and trade secrets were present. Through the course of the investigation, I uncovered a number of interesting items that would lead to the conclusion that the employee should be charged with multiple felonies.

For the second part, I chose to analyze a forensic tool to determine its suitability for computer forensic work. I selected X-Ways Forensics, specifically their disk cloning and hashing ability. By comparing its functions with other common tools that perform similar functions, I was able to determine that, while there were subtleties with how the tool functioned, overall it performed satisfactorily.

Part I – Analyze an Unknown Image

The following scenario was presented.

Robert John Leszczynski, Jr., is employed by Ballard Industries, a designer of fuel cell batteries which produces specialized batteries used around the world by thousands of companies. Robert is assigned as the lead process control engineer for the project.

After several successful years of manufacturing and distributing a relatively new fuel cell battery, which is used in many applications, Ballard industries notices that many of their clients are no longer re-ordering from them.

After making several calls the vice president of sales determines that one of Ballard's major competitors, Rift, Inc., has been receiving the new orders for the same fuel cell battery which was once unique to Ballard. A full blown investigation ensues.

The investigation has not turned up very much. It is apparent that Rift, Inc. somehow has received proprietary information from Ballard industries. Ballard industries keeps a customer database of all its clients and it is feared that that information somehow got out along with other proprietary data.

The only thing out of the ordinary that has turned up is a floppy disk that was being taken out of the R&D labs by Robert Leszczynski on 26 April 2004 at approximately 4:45 pm MST, which is against company policy. The on staff security guard seized the floppy disk from Robert's briefcase and told Robert he could retrieve it from the security administrator.

The security administrator, David Keen, has asked you to analyze the floppy disk and provide a report of your findings prior to returning it to Robert. He provides you with a chain of custody form with the following information:

- * Tag# fl-260404-RJL1
- * 3.5 inch TDK floppy disk
- * MD5: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
- * fl-260404-RJL1.img.gz

Before going into details of the analysis, presented here are some of the tools I used for the benefit of those who may not be familiar with them.

Tools

1. HashCalc, version 2.01. a GUI driven Windows program that is capable of generating up to 13 different styles of hashes at the time of this paper's writing, including MD5 and SHA-1 (Slavasoft, 2004). HashCalc is a free tool downloadable from <http://www.slavasoft.com/hashcalc/>.
2. md5sum – version 5.2.1. Written by Ulrich Drepper and Scott Miller. Linux program that will compute the MD5 sum of a specified file.
3. diff – version 2.8.1, written by Paul Eggert, Mike Haertel, David Hayes, Richard Stallman, and Len Tower. Linux program that takes two files and attempts to determine what differences (if any) exist. Useful in determining small differences between large files.
4. Autopsy – Version 2.01. An open source Linux tool designed to assist forensic investigators with performing forensic analysis, and is based on the forensic tool kit “The Sleuth Kit”. It provides a convenient and efficient graphic front end for a number of command line tools, and helps alleviate some of the headache in performing forensic work (Carrier, 2005). See <http://www.sleuthkit.org/autopsy/desc.php>.
5. fsstat – The Sleuth Kit ver 1.70, Linux command line tool used to

gather system state information

6. nslookup – Windows command line tool used to gather DNS information (also available on Linux, but wasn't used within this practical).

7. www.dnsstuff.com – free website that allows users to query Internet DNS servers and related information pertaining to domain names and IP addresses.

8. frhed – version 1.0.156 beta 1. Pabs – free windows hex editor (Kibria, 2005). See <http://www.kibria.de/frhed.html>

In addition, the following machines, isolated from any network, were used during the analysis

1. Windows Machine – Dell Optiplex GX270 2.6 GHZ running Windows XP SP 1 with 1 GB RAM.
2. Linux Machine – Dell Optiplex GX260 1.0 GHZ running RedHat Fedora Core 2 with 512 MB RAM.

The assignment indicates that a chain of custody form has been filled out and is attached to the data. While this is only a hypothetical scenario and the chain of custody form exists only within the context of the assignment, it's important to note that, were this an actual event, I would complete my portion of the chain of custody, indicating I had taken ownership of the evidence.

The first task was to transfer the floppy image to a network-isolated Linux machine I'd set up for purposes of performing forensic analysis. I then wanted to verify that the image had transferred over without being modified and was still forensically sound.

```
[root@LinuxForensics giac]# cat keen.md5
d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
[root@LinuxForensics giac]# md5sum fl-260404-RJL1.img > floppy.md5
[root@LinuxForensics giac]# cat floppy.md5
d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
[root@LinuxForensics giac]# diff keen.md5 floppy.md5
[root@LinuxForensics giac]#
```

I confirmed they matched by running the Linux command “diff” against the expected output provided by Mr. Keen and the md5sum result I ran. At this point, I assumed that the file I had had not been modified in transit.

The next step was to get the file loaded into Autopsy. Before importing the file into Autopsy, I needed to confirm the file format of the image.

Judging from the size of the image file of 1474560, it appears it might be an image of a 3.5" 1.44MB diskette.

```
-rwxr-xr-x  1 root root 1474560 Nov  3 09:30 fl-260404-  
RJL1.img
```

Examining the file in a hex editor, at byte offset 0000:0036, I see the ASCII "FAT12", and at byte offset 0000:0003, I see "mkdosfs".

```
000000  eb 3c 90 6d 6b 64 6f 73 66 73 00 00 02 01 01 00  <.mkdosfs.....  
000010  02 e0 00 38 0b f0 09 00 12 00 02 00 00 00 00 00  .à.8.ö.....  
000020  00 00 00 00 00 00 29 14 ed 8b 40 52 4a 4c 20 20  .....).i.@RJL  
000030  20 20 20 20 20 20 46 41 54 31 32 20 20 20 0e 1f  ..FAT12..
```

To confirm this is indeed a FAT12 image file, I run fsstat against the image file specifying the FAT12 file system.

```
[root@LinuxForensics giac]# fsstat -f fat12 fl-260404-RJL1.img  
FILE SYSTEM INFORMATION
```

```
-----  
File System Type: FAT
```

```
OEM Name: mkdosfs  
Volume ID: 0x408bed14  
Volume Label (Boot Sector): RJL  
Volume Label (Root Directory): RJL  
File System Type Label: FAT12
```

```
Sectors before file system: 0
```

```
File System Layout (in sectors)  
Total Range: 0 - 2871  
* Reserved: 0 - 0  
** Boot Sector: 0  
* FAT 0: 1 - 9  
* FAT 1: 10 - 18  
* Data Area: 19 - 2871  
** Root Directory: 19 - 32  
** Cluster Area: 33 - 2871
```

```
META-DATA INFORMATION
```

```
-----  
Range: 2 - 45426  
Root Directory: 2
```

```
CONTENT-DATA INFORMATION
```

```
-----  
Sector Size: 512  
Cluster Size: 512  
Total Cluster Range: 2 - 2840
```

```
FAT CONTENTS (in sectors)  
-----
```

```
105-187 (83) -> EOF
188-250 (63) -> EOF
251-316 (66) -> EOF
317-918 (602) -> EOF
919-1340 (422) -> EOF
1341-1384 (44) -> EOF
```

I create a new case and I mount the image with the following parameters defined in Autopsy.

ADD A NEW IMAGE

1. **Location:** The full path (starting with /) to the raw file system image.

2. **Import Method:** The image can be imported into the Autopsy Evidence Locker from its current location by making a symbolic link, by copying it, or by moving it. Note that if a system failure occurs during the move, then the image could become corrupt.
☐ Symlink ☒ Copy ☐ Move

3. **File System Type:** Specify the type of file system.

4. **Mount Point:** The directory or drive where the file system was mounted in the original suspect system (i.e. C:\ for Windows or /usr/ for UNIX). Not needed for swap or raw file system types.
 other:

5. **Data Integrity:** An MD5 hash can be used to verify the integrity of the file system image.
☒ Calculate the hash value for this image.
☐ Ignore the hash value for this image.
☐ Add the following hash value for this image:

☒ Verify MD5 After Importing?

ADD IMAGE **CANCEL** **HELP**

Autopsy's MD5 summary indicates the copy was successful.


```
Copying /images/giac/fl-260404-RJL1.img to /forensics/test2/floppy1/images/fl-260404-RJL1.img
```

```
Calculating MD5 of images/fl-260404-RJL1.img (this could take a while)
```

```
Current MD5: D7641EB4DA871D980ADBE4D371EDA2AD
```

```
Image: /images/giac/fl-260404-RJL1.img added to config file as images/fl-260404-RJL1.img
```

OK

ADD IMAGE

Once the image is mounted in Autopsy, I navigate to Keyword Search -> Extract Strings and Extract Unallocated, after which I extract unallocated strings. I then begin to parse through the two resultant files, *fl-260404-RJL1.img.str* and *fl-260404-RJL1.img.dls.str*, by opening the files with “less” and scrolling through the results.

```
[root@LinuxForensics output]# d
total 952
drwxr-xr-x  2 root root  4096 Feb 10 12:27 .
drwxr-xr-x  7 root root  4096 Feb 10 12:27 ..
-rw-r--r--  1 root root 798208 Feb 10 12:22 fl-260404-RJL1.img.dls
-rw-r--r--  1 root root   7254 Feb 10 12:27 fl-260404-RJL1.img.dls.str
-rw-r--r--  1 root root 144162 Feb 10 12:22 fl-260404-RJL1.img.str
-rw-r--r--  1 root root    199 Feb 10 12:27 md5.txt

[root@LinuxForensics output]# less fl-260404-RJL1.img.str
```

Strings Analysis

A number of items jump out within the first few pages. First, I notice the presence of a number of .doc files, presumably the policy files. There is also a possible *_ndex.htm* web page file, which Autopsy shows has been deleted.

All Deleted Files

Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID	META
r / r	a:\CamShell.dll (AMSHELL.DLL)	2001.02.03 19:44:16 (MST)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:18 (MDT)	36864	0	0	<u>5</u>
r / r	a:_ndex.htm	2004.04.23 10:53:56 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:47:36 (MDT)	727	0	0	<u>28</u>

Deleted Files according to Autopsy

There are a number of text strings indicated HTML tags, which I’m guessing at this point are from the .htm file, but they point to a shockwave file “*ballard.swf*”. I note this as something that will later need to be revisited.

Further digging into the strings file shows instances of the following strings:

SheCamouflageShell
ShellExt
VB5!
CamShell
pwReserved

This worries me that there is a program that is designed to possibly mask a shell prompt, and that it likely is a Visual Basic compiled program. Since “cmd.exe” is the default windows command shell, it stands to reason that I might be looking for some kind of Windows command terminal program. I also wonder if there is some kind of password lock on this program from the “pwReserved” mention. I again note this as something to be revisited.

The strings file also show what appears to be text from various information security policy documents, relating to passwords, lab protocols, remote access rules, etc. The strings following these documents contain similar strings...

Microsoft Word 10.0
Ballard
MSWordDoc

...etc. This suggests these documents are in fact Microsoft Word documents. I also notice that, in between some of the policy documents, there are significant amounts of random character strings like what you might find in a binary or non-ASCII file. While I would anticipate some of those because Word documents do contain special non-ASCII formatting characters, there seem to be too many.

At this point, I’ve written down a number of questions I need to have answered:

1. What is the significance of the _ndex.htm file?
 - a. What is “ballard.swf” and is it present in some form on the diskette?
 - b. Can I confirm the .htm file was deleted, and if so, was this file deleted for a reason?
2. What is “CamShell”?
3. Why is there lots of non-ASCII content in between some of the Word Documents?

Timeline Analysis

Since unallocated space had already been examined by Autopsy, creating the timeline didn’t require much extra effort. Attached is the resultant output of the timeline as generated by Autopsy.

```

Sat Feb 03 2001 19:44:16 36864 m.. -/rwxxrwxrwx 0 0 5 /CamShell.dll (_AMSHLL.DLL) (deleted)
36864 m.. -/rwxxrwxrwx 0 0 5 <fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
Thu Apr 22 2004 16:31:06 33423 m.. -/rwxxrwxrwx 0 0 17 /Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
32256 m.. -/rwxxrwxrwx 0 0 13 /Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Fri Apr 23 2004 10:53:56 727 m.. -/rwxxrwxrwx 0 0 28 /_ndex.htm (deleted)
727 m.. -/rwxxrwxrwx 0 0 28 <fl-260404-RJL1.img-_ndex.htm-dead-28>
Fri Apr 23 2004 11:54:32 215895 m.. -/rwxxrwxrwx 0 0 23 /Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26 307935 m.. -/rwxxrwxrwx 0 0 20 /Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50 22528 m.. -/rwxxrwxrwx 0 0 27 /Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10 42496 m.. -/rwxxrwxrwx 0 0 9 /Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Sun Apr 25 2004 00:00:00 0 .a. -/rwxxrwxrwx 0 0 3 /RJL (Volume Label Entry)
Sun Apr 25 2004 10:53:40 0 m.c -/rwxxrwxrwx 0 0 3 /RJL (Volume Label Entry)
Mon Apr 26 2004 00:00:00 32256 .a. -/rwxxrwxrwx 0 0 13 /Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
22528 .a. -/rwxxrwxrwx 0 0 27 /Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
42496 .a. -/rwxxrwxrwx 0 0 9 /Information_Sensitivity_Policy.doc (INFORM~1.DOC)
727 .a. -/rwxxrwxrwx 0 0 28 /_ndex.htm (deleted)
215895 .a. -/rwxxrwxrwx 0 0 23 /Remote_Access_Policy.doc (REMOTE~1.DOC)
727 .a. -/rwxxrwxrwx 0 0 28 <fl-260404-RJL1.img-_ndex.htm-dead-28>
36864 .a. -/rwxxrwxrwx 0 0 5 <fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
33423 .a. -/rwxxrwxrwx 0 0 17 /Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
36864 .a. -/rwxxrwxrwx 0 0 5 /CamShell.dll (_AMSHLL.DLL) (deleted)
307935 .a. -/rwxxrwxrwx 0 0 20 /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:18 36864 .c -/rwxxrwxrwx 0 0 5 /CamShell.dll (_AMSHLL.DLL) (deleted)
36864 .c -/rwxxrwxrwx 0 0 5 <fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
Mon Apr 26 2004 09:46:20 42496 .c -/rwxxrwxrwx 0 0 9 /Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22 32256 .c -/rwxxrwxrwx 0 0 13 /Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24 33423 .c -/rwxxrwxrwx 0 0 17 /Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26 307935 .c -/rwxxrwxrwx 0 0 20 /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36 215895 .c -/rwxxrwxrwx 0 0 23 /Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44 22528 .c -/rwxxrwxrwx 0 0 27 /Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36 727 .c -/rwxxrwxrwx 0 0 28 /_ndex.htm (deleted)
727 .c -/rwxxrwxrwx 0 0 28 <fl-260404-RJL1.img-_ndex.htm-dead-28>

```

By virtue of this being a FAT12 floppy disk, it should be noted that access times are not recorded, only access dates, modified times and dates, and change times and dates (Fox, 2002). Detailed explanation of the FAT12 file structure can be found at <http://home.freeuk.net/foxy2k/disk/disk6.htm> and <http://www.mega-tokyo.com/osfaq2/index.php/FAT12%20document>.

Based on the above information, I make some educated guesses.

1. CamShell.dll has been on this floppy as recently as Feb. 3, 2001. Presumably, if any bad stuff has been happening, its been happening for a while.
2. Two files with similar names, "Internal_Lab_Policy.doc" and "Internal_Lab_Policy1.doc", were modified on Apr. 22, 2004 at 16:31:06, but they each have slightly differing sizes. Presumably some script/program was run modify the content of each file?
3. Various other .doc files were transferred to the diskette the next day, and time stamps vary slightly, probably not scripted but likely manually copied.
4. According to this timeline, the volume label is accessed at exactly midnight on Apr. 25. As I stated before, access times are not recorded. Autopsy will arbitrarily fill in 00:00:00 as access times in all cases for FAT12. Since the volume label is both modified and created at 10:53am on Apr. 25, it's reasonable to assert the volume label was accessed on or after this time on Apr. 25
5. A bunch of files get accessed at midnight on Apr. 26. Again, since access times are not recorded anywhere on the floppy image, 00:00:00 is filled in arbitrarily. Two deleted files at inodes 5 and 28 show accessed

also.

6. Finally, at around 9:46am on Apr. 26, a number of these same files were changed. Change indicates information about the inode of the file was adjusted. CamShell.dll and index.htm are shown as changed at around this time, and its at this time that I suspect these files are truly deleted off the diskette.

File Analysis

I was able to recover the following files from the diskette (md5sums included)...

99c5dec518b142bd945e8d7d2fad2004	Information_Sensitivity_Policy.doc (INFORM~1.DOC)
e0c43ef38884662f5f27d93098e1c607	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
b9387272b11aea86b60a487fbdc1b336	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
ac34c6177ebdc4f4adc41f0e181belbc	Password_Policy.doc (PASSWO~1.DOC)
5b38dlac1f94285db2d2246d28fd07e8	Remote_Access_Policy.doc (REMOTE~1.DOC)
f785ba1d99888e68f45dabeddb0b4541	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)

According to autopsy, two files show as deleted. I was able to recover those as well...

6462fb3acca0301e52fc4ffa4ea5eff8	CamShell.dll (_AMSHLL.DLL)
17282ea308940c530a86d07215473c79	_ndex.htm

I was unable to recover user/group information about these files. The following snip from autopsy shows details about each file, size, inode location, etc.

Current Directory: /									
ADD NOTE GENERATE MD5 LIST OF FILES									
DEL	Type	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID	META
✓	r/r	index.htm	2004.04.23 10:53:56 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:47:36 (MDT)	727	0	0	28
	r/r	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)	2004.04.23 14:10:50 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:44 (MDT)	22528	0	0	27
✓	r/r	CamShell.dll (_AMSHLL.DLL)	2001.02.03 19:44:16 (MST)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:18 (MDT)	36864	0	0	5
	r/r	Information_Sensitivity_Policy.doc (INFORM~1.DOC)	2004.04.23 14:11:10 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:20 (MDT)	42496	0	0	9
	r/r	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)	2004.04.22 16:31:06 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:24 (MDT)	33423	0	0	17
	r/r	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)	2004.04.22 16:31:06 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:22 (MDT)	32256	0	0	13
	r/r	Password_Policy.doc (PASSWO~1.DOC)	2004.04.23 11:55:26 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:26 (MDT)	307935	0	0	20
	r/r	Remote_Access_Policy.doc (REMOTE~1.DOC)	2004.04.23 11:54:32 (MDT)	2004.04.26 00:00:00 (MDT)	2004.04.26 09:46:36 (MDT)	215895	0	0	23
	r/r	RJL (Volume Label Entry)	2004.04.25 10:53:40 (MDT)	2004.04.25 00:00:00 (MDT)	2004.04.25 10:53:40 (MDT)	0	0	0	3

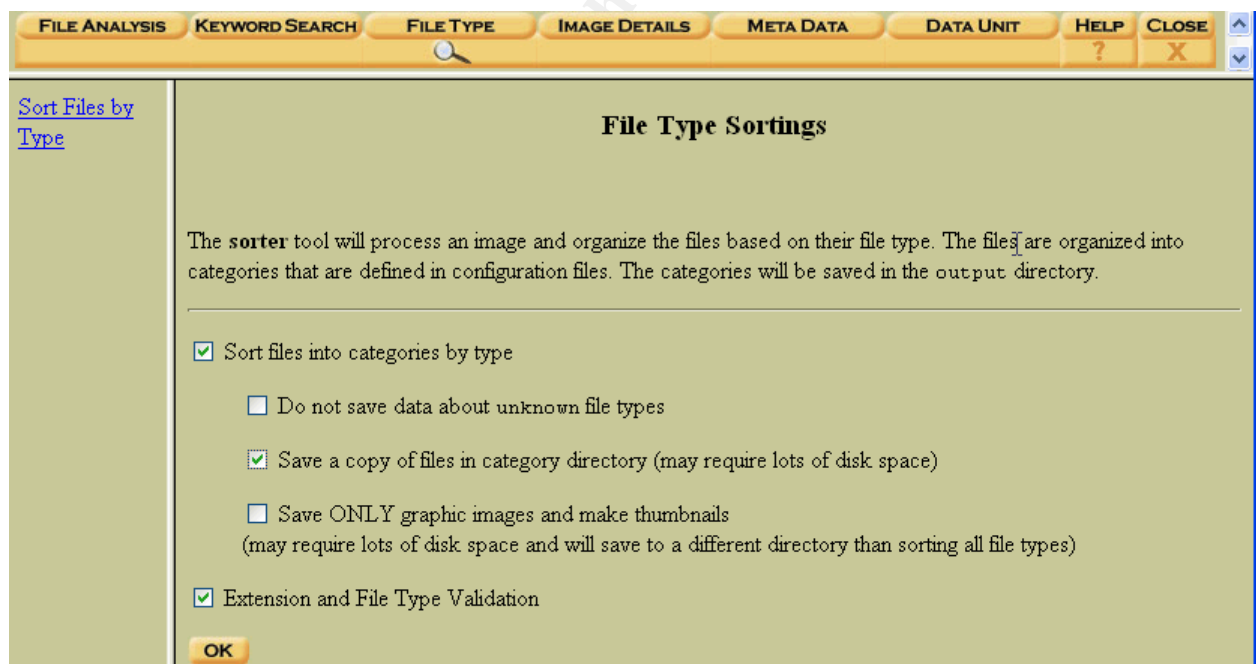
It's interesting to me that there is no mention of a "ballard.swf" anywhere at this

point, so the first thing I want to do is find if its hiding anywhere on the diskette.

I do a quick Google search looking for “swf magic file forensics” and the first hit takes me to http://www.garykessler.net/library/file_sigs.html. A file’s “magic” refers to characteristic headers one would find in any given type of file. In this case, it says that shockwave files have a characteristic 46h 57h 53h, or translated to ASCII, “FWS”. I pull the diskette image up in khedit and do searches for that combination of characters. Nothing is found. I run grep against the strings output file looking for swf, shockwave, and ballard.swf, and nothing more than what I already know comes up. At this point, I’m going to skip the mysteries of index.htm and ballard.swf and move on.

I decide to run the file type sorting utility within Autopsy by selecting: File Type -> “Sort Files by Type”, which is really just running “sorter” on the image. What “sorter” does is runs ‘fls -r’ against each file, and then runs ‘icat’ against it. The fls tool examines an image at the file level and obtains names of detected files, as well as deleted files, and their correlating inodes. The icat tool takes a file at a given inode and exports it, thus recovering the file.

I save copies of detected files for later perusal.



Attached is a summary of the findings.

sorter output

Images

- ♦ /forensics/GLAC-GCFA-Certification/floppy1/images/fl-260404-RJL1.img

Files (14)

- ♦ Allocated (9)
- ♦ Unallocated (5)

Files Skipped (4)

- ♦ Non-Files (4)
- ♦ 'ignore' category (0)

Extensions

- ♦ [Extension Mismatches](#) (1)

Categories (10)

- ♦ archive (0)
- ♦ audio (0)
- ♦ compress (0)
- ♦ crypto (0)
- ♦ data (0)
- ♦ disk (0)
- ♦ [documents](#) (6)
- ♦ exec (0)
- ♦ images (0)
- ♦ system (0)
- ♦ [text](#) (4)
- ♦ unknown (0)
- ♦ video (0)

There are 6 document files extracted and some text files as well. Note the presence of an extension mismatch. This correlates to Camshell.dll, on inode 5. Closer examination of the recovered file shows there is HTML text within the DLL, not expected.

I open up the sorter-exported file (which I refer to in this document as “_AMSHLL.DLL.inode5”) in khedit and scroll through the contents. I note the presence of the same HTML at the beginning of the file as was in index.htm. Further into the file looks like binary information, but some interesting human-readable content is present as well.

007220	6f 00 00 00	94 03 00 00	01 00 30 00	34 00 30 00	o.....0.4.0.
007230	39 00 30 00	34 00 42 00	30 00 00 00	64 00 4c 00	9.0.4.B.0...d.L.
007240	01 00 43 00	6f 00 6d 00	6d 00 65 00	6e 00 74 00	..Comment.
007250	73 00 00 00	68 00 74 00	74 00 70 00	3a 00 2f 00	s...h.t.t.p.../
007260	2f 00 77 00	77 00 77 00	2e 00 63 00	61 00 6d 00	/www.cam.
007270	6f 00 75 00	66 00 6c 00	61 00 67 00	65 00 2e 00	ou.flage.
007280	66 00 72 00	65 00 65 00	73 00 65 00	72 00 76 00	free.serv.
007290	65 00 2e 00	63 00 6f 00	2e 00 75 00	6b 00 00 00	e...co.uk.
0072a0	54 00 32 00	01 00 43 00	6f 00 6d 00	70 00 61 00	T.2...Comp.a.
0072b0	6e 00 79 00	4e 00 61 00	6d 00 65 00	00 00 00 00	ny.Name....
0072c0	54 00 77 00	69 00 73 00	74 00 65 00	64 00 20 00	T.wisted.
0072d0	50 00 65 00	61 00 72 00	20 00 50 00	72 00 6f 00	Pear..Pro.
0072e0	64 00 75 00	63 00 74 00	69 00 6f 00	6e 00 73 00	ductions.
0072f0	00 00 00 00	b0 00 88 00	01 00 46 00	69 00 6c 00Fil.
007300	65 00 44 00	65 00 73 00	63 00 72 00	69 00 70 00	e.Descrip.
007310	74 00 69 00	6f 00 6e 00	00 00 00 00	4b 00 65 00	tion....Ke.
007320	65 00 70 00	73 00 20 00	66 00 69 00	6c 00 65 00	eps...file.
007330	73 00 20 00	63 00 6f 00	6e 00 74 00	61 00 69 00	s...contai.
007340	6e 00 69 00	6e 00 67 00	20 00 73 00	65 00 6e 00	ning...sen.
007350	73 00 69 00	74 00 69 00	76 00 65 00	20 00 69 00	sitive...i.
007360	6e 00 66 00	6f 00 72 00	6d 00 61 00	74 00 69 00	nformati.
007370	6f 00 6e 00	20 00 73 00	61 00 66 00	65 00 20 00	on...safe.
007380	66 00 72 00	6f 00 6d 00	20 00 70 00	72 00 79 00	from...pry.
007390	69 00 6e 00	67 00 20 00	65 00 79 00	65 00 73 00	ing...eyes.
0073a0	2e 00 00 00	cc 00 a8 00	01 00 4c 00	65 00 67 00	...l...Leg.
0073b0	61 00 6c 00	43 00 6f 00	70 00 79 00	72 00 69 00	al.Copyri.
0073c0	67 00 68 00	74 00 00 00	43 00 6f 00	70 00 79 00	ght...Copy.
0073d0	72 00 69 00	67 00 68 00	74 00 20 00	28 00 63 00	right... (c.
0073e0	29 00 20 00	32 00 30 00	30 00 30 00	2d 00 32 00)...2.0.0.0.-2.
0073f0	30 00 30 00	31 00 20 00	62 00 79 00	20 00 54 00	0.0.1...by...T.
007400	77 00 69 00	73 00 74 00	65 00 64 00	20 00 50 00	wisted...P.
007410	65 00 61 00	72 00 20 00	50 00 72 00	6f 00 64 00	ear...Prod.
007420	75 00 63 00	74 00 69 00	6f 00 6e 00	73 00 2c 00	uctions...
007430	20 00 41 00	6c 00 6c 00	20 00 72 00	69 00 67 00	...All...rig.
007440	68 00 74 00	73 00 20 00	72 00 65 00	73 00 65 00	hts...rese.
007450	72 00 76 00	65 00 64 00	20 00 77 00	6f 00 72 00	rved...wor.
007460	6c 00 64 00	77 00 69 00	64 00 65 00	2e 00 00 00	ld.wide....
007470	38 00 16 00	01 00 50 00	72 00 6f 00	64 00 75 00	8....Produ.
007480	63 00 74 00	4e 00 61 00	6d 00 65 00	00 00 00 00	ct.Name....
007490	43 00 61 00	6d 00 6f 00	75 00 66 00	6c 00 61 00	Cam.ou fla.
0074a0	67 00 65 00	00 00 00 00	34 00 14 00	01 00 46 00	ge....4....F.
0074b0	69 00 6c 00	65 00 56 00	65 00 72 00	73 00 69 00	ile.Versi.
0074c0	6f 00 6e 00	00 00 00 00	31 00 2e 00	30 00 31 00	on....1....0.1.
0074d0	2e 00 30 00	30 00 30 00	31 00 00 00	38 00 14 00	...0.0.0.1...8...
0074e0	01 00 50 00	72 00 6f 00	64 00 75 00	63 00 74 00	...Produ.ct.
0074f0	56 00 65 00	72 00 73 00	69 00 6f 00	6e 00 00 00	Vers.ion...
007500	31 00 2e 00	30 00 31 00	2e 00 30 00	30 00 30 00	1...0.1...0.0.0.
007510	31 00 00 00	34 00 12 00	01 00 49 00	6e 00 74 00	1...4....Int.
007520	65 00 72 00	6e 00 61 00	6c 00 4e 00	61 00 6d 00	ernal.Nam.
007530	65 00 00 00	43 00 61 00	6d 00 53 00	68 00 65 00	e...C.am.S.he.

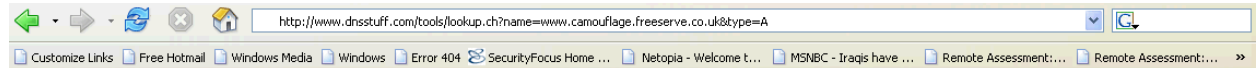
Excerpt of recovered _AMSHLL.DLL.inode5

Useful bits of information here are located in a Comment section of the file.

- There's a URL of <http://www.camouflage.freeseve.co.uk>.
- There's a company name "Twisted Pear Productions".
- File Description is: Keeps files containing sensitive information safe from prying eyes.
- Copyright: 2000-2001
- Product Name: Camouflage
- File Version: 1.01.0001
- Production Version: 1.01.0001

On a separate, internet connected workstation, I do an nslookup of the referenced URL, but no information is returned. I then navigate to

www.dnsstuff.com and do a few additional lookups as confirmation. See screenshots below.



DNS Lookup: www.camouflage.freemove.co.uk A record

Generated by www.DNSstuff.com

How I am searching:
Searching for A record for www.camouflage.freemove.co.uk at d.root-servers.net: Got referral to NSA.NIC.uk. [took 71 ms]
Searching for A record for www.camouflage.freemove.co.uk at NSA.NIC.uk.: Got referral to pridns1.svr.pol.co.uk. [took 60 ms]
Searching for A record for www.camouflage.freemove.co.uk at pridns1.svr.pol.co.uk.: Reports that no A records exist. [took 219 ms]

Answer:
No A records exist for www.camouflage.freemove.co.uk. [Neg TTL=14400 seconds]

Details:
pridns1.svr.pol.co.uk. (an authoritative nameserver for freemove.co.uk.) says that there are no A records for www.camouflage.freemove.co.uk.
The E-mail address in charge of the freemove.co.uk. zone is: hostmaster@planet.net.uk.

To see the reverse DNS traversal, to make sure that all DNS servers are reporting the correct results, you can [Click Here](#).

The domain freemove.co.uk appears to be a hosting domain, but the camouflage.freemove.co.uk subdomain doesn't seem to exist anymore.



WHOIS results for camouflage.freemove.co.uk

Generated by www.DNSstuff.com

Found WHOIS server for .uk: whois.nic.uk. Looking up.

No match for "camouflage.freemove.co.uk".

--

(c) Nominet UK 1996 - 2004

I do a number of Google searches, looking for anything related to camouflage and hiding files, and I'm able to locate what appears to be a mirror site of the original www.camouflage.freemove.co.uk site located at <http://camouflage.unfiction.com/>. It appears that "camouflage" is a stegonographic program used to hide and encrypt files within files. Verbiage from the website as follows...

What is 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 emailed 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 Windows 95, Windows 98, Windows ME, Windows NT and Windows 2000, and is simple to install and use. (Twisted Pear Productions, 2005)

In order to use Camouflage, two files are required: first, the file you want hidden. Second, the file you want the hidden file embedded into. At the time you run camouflage, you have the option to define a password.

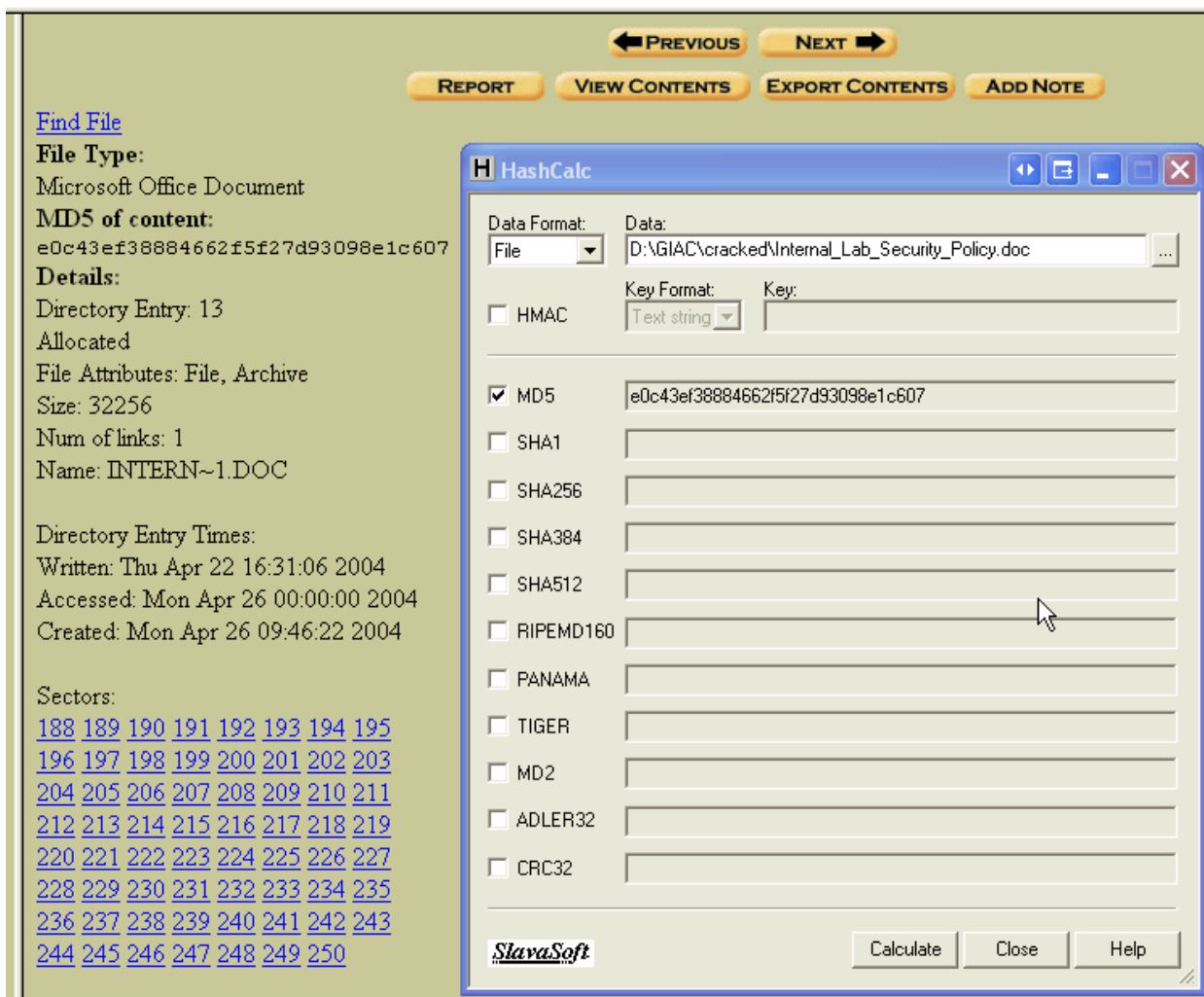
I download a copy of the camouflage program, version 1.2.1 and install it on a non-networked Windows XP workstation. If the above hex from the CamShell.dll is any indication, the file version is 1.01.0001. If I examine the hex of the CamouflageShell.dll that comes with it, it shows the following...

007410	6d 00 6f 00	15 00 66 00	6c 00 61 00	67 00 65 00	m.o.u.f.l.a.g.e.
007420	00 00 00 00	28 00 0a 00	01 00 46 00	69 00 6c 00F.i.l.
007430	65 00 56 00	65 00 72 00	73 00 69 00	6f 00 6e 00	e.V.e.r.s.i.o.n.
007440	00 00 00 00	31 00 2e 00	30 00 30 00	00 00 00 00	...1...0.0.....

Presumably, then, even though the website claims its version 1.2.1, I either have an older version of the Camouflage software than what was used in this hack or the version number is not reliable.

I copy the 6 document files exported from sorter onto the Windows box. I then attempt to uncamouflage each of the six document files, opting for no password.

One file yields success, "Internal_Lab_Security_Policy.doc". I'm able to extract out two files from this single file, one labeled as "Internal_Lab_Security_Policy.doc" and one labeled as "Opportunity.txt". The first one seems to be the same document as what resides on inode 13 labeled as "Internal_Lab_Security_Policy1.doc". A quick run of md5 hashes on both files shows that they are the same. The screenshot below is a display of two programs, the left one is autopsy showing the md5 hash of the file as it is on the diskette image. On the right is the md5 hash of the uncamouflaged file.



Because I am not sure yet if "Opportunity.txt" is truly a text file, I decide to open it up inside the frhed windows hex editor, and note the lack of any strange binary kind of data, and that it appears to be a text file. Output is as 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

Extract of Opportunity.txt

None of the other files would yield anything when I tried to uncamouflage them with no password. However, the Camouflage program author indicates that Camouflage will not tell you whether a file is a true camouflage file or not if you put in the incorrect password. Thus, I'm encouraged because one of them

worked, and, if we are to read the resultant message literally, would suggest there was more to find hidden on this diskette, specifically: one database, and more than one of some kind of schematic. My suspicion also is that the statement, “They are available as we discussed – “First Name”” suggests the manner in which they can be extracted. Perhaps this is a password clue.

At this point I’m curious if there’s some way I can tell if a file has a camouflaged file within it. I suspect that if I can find the hex difference between the combined file .doc and the extracted .doc, I will find what I’m looking for. To accomplish this, I bring up each file within khexedit and perform an “export” function, which will dump the hex code to standard ASCII format. I then run the linux command “diff” against themselves, and it dumps a section that is different, starting at byte offset 0x7e00. Attached is a snip of the first few lines it shows.

```
007e00 | 20 00 b9 28 | c4 01 60 38 | b8 73 75 29 | c4 01 e0 36 | .¹(Ä.´8,su)Ä.à6
007e10 | 98 ba b9 28 | c4 01 30 ff | 7b 7f 38 01 | 00 00 4b b5 | .²¹(Ä.0ÿ{.8...Kµ
007e20 | 1b 4f 2c d1 | 7d 8d 8d a6 | d1 02 00 1b | f1 93 e9 17 | .O.N}...|N...ñ.é.
007e30 | 25 25 92 92 | 10 74 d4 4c | b8 5e eb 40 | 93 94 c2 94 | %%'´.tÔL,^e@..Ä.
007e40 | 22 a0 27 6e | 6f ae 9e f5 | 6e 0f 59 00 | 4d 6f 3b ff | " 'nq@.Ön.Y.Mo;ÿ
```

Internal_Lab_Security_Policy.doc
(Inode 17)

I choose to take the first two bytes and run a search against the other files. I show two files that have similar patterns. Extracts are as follows.

```
009c00 | 20 00 46 29 | c4 01 30 48 | e8 d0 75 29 | c4 01 70 af | .F)Ä.à0HèDu)Ä.p~
009c10 | 1f b6 5c 29 | c4 01 70 a0 | b3 27 07 6e | 00 00 fd 4d | .¶\)\Ä.p ³'.n..ýM
```

Password_Policy.doc
(Inode 20)

```
007800 | 20 00 46 29 | c4 01 e0 b6 | 71 d3 75 29 | c4 01 c0 3c | .F)Ä.à¶qÓu)Ä.Ä<
007810 | d8 fe 5c 29 | c4 01 60 49 | 81 07 00 d0 | 02 00 02 94 | 0p\)\Ä.´I...Ð....
```

Remote_Access_Policy.doc
(Inode 23)

There are a great many similarities here. I noticed that the 3rd and 4th bytes are identical, each 46 29. I hypothesize that if a blank password were denoted with b9 28 as shown in the first document, if I changed those bytes in the other two, would that clear out the need for a password as far as camouflage were concerned? I adjusted the necessary bytes, but it yielded nothing. Could it be that the password was the same among all the files?

I then hypothesized, “What if the clue ‘First Name’ were the password”? I tried entering the string “First Name” as the password, no luck. I changed around capitalization, no luck. I added quotes, no luck. I tried it without the space between the words, no luck.

I then hypothesized, suppose it were Robert’s first name. I tried “Robert”, “Rob”,

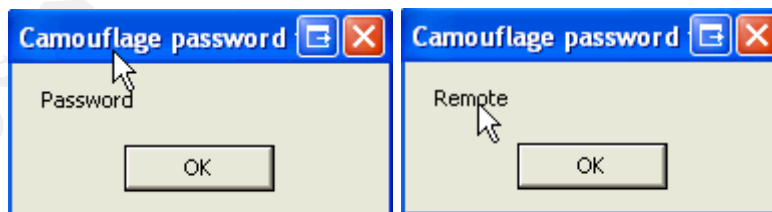
“Bob”, I even tried “David” from David Keen, some varieties of upper/lower case. No luck.

I then hypothesized, suppose Robert means the first file name showing up on the diskette. Alphabetically, that would be “Acceptable” from the “Acceptable_Encryption_Policy.doc”. No luck.

I then theorized that the name might be buried somewhere within the policy files themselves. I performed string searches in each of the word documents looking for either “first” or “name” and nothing stood out. I examined the meta data within the word documents, and tried variations of “ballard”, “RJL”, “root”, and “cisco”. No luck. I tried “password”. Again, nothing. I tried running string searches against the entire diskette image, again looking for non-case specific “first” or “name”. Nothing.

At this point, I decided to stop guessing and see if someone had figured out how to break passwords that were camouflaged. I ran a number of searches on www.astalavista.com, and I found a program, written by a group calling themselves the “Desperate Turk Crackers Group”. I copied the .zip file to a third isolated Windows XP machine using a USB keychain drive and attempted to decompress it, but received winzip CRC errors. I tried correcting those CRC errors using “Advanced Zip Repair” which is supposed to fix such errors. I now was able to decompress the files, but when I attempted to run them, it caused the application to crash. This line of investigation was a dead end.

More searching on the Internet yielded what I was looking for. An excellent paper, written by Guillermi to and located <http://guillermi2.net/stegano/camouflage>, goes into great detail on the weaknesses of the camouflage encryption and password protection (Guillermi to, 2002). He also provides tools for cracking the password on camouflaged files. I obtain the tool and run it. It prompts me to select a file to run the tool against, and when I click “Open”, a small window appears with what it believes the password to be.



The tool indicates that the passwords are “Password” and “Remote”, each respective of their file, Password_Policy.doc and Remote_Access_Policy.doc. I attempt to uncamouflage those files using my newly acquired passwords, and was able to successfully extract the hidden files.

From *Password_Policy.doc*, I obtain the files (md5sum's included)...

```
9da5d4c42fdf7a979ef5f09d33c0a444
/mnt/floppy/Hydrocarbon%20fuel%20cell%20page2.jpg
e5066b0fb7b91add563a400f042766e4 /mnt/floppy/Password_Policy.doc
864e397c2f38ccfb778f348817f98b91 /mnt/floppy/pem_fuelcell.gif
5e39dcc44acccdca7bba0c15c6901c43 /mnt/floppy/PEM-fuel-cell-large.jpg
```

From *Remote_Access_Policy.doc*

```
c3a869ff6b71c7be3eb06b6635c864b1 /mnt/floppy/cat.mdb
2afb005271a93d44b6a8489dc4635c1c /mnt/floppy/Remote_Access_Policy.doc
```

I ran *mac_daddy.pl* against these newly uncloaked files, output shown below. The change times show when I copied the files over, thus can be disregarded. However, the modified times are accurate. Note that it says the access times are the same as the modified times. This is an eccentricity of *mac_daddy.pl*; we saw earlier where *autopsy* recorded access times as midnight because access times are not recorded anywhere on the floppy image. *Mac_daddy.pl*'s behavior is to assert the access time at the time of modification

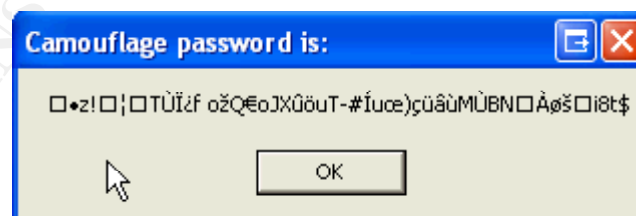
```
Apr 22 2004 16:31:06 32256 ma. -rwxr-xr-x root root /mnt/floppy/Internal_Lab_Security_Policy.doc
Apr 23 2004 10:15:18 30264 ma. -rwxr-xr-x root root /mnt/floppy/pem_fuelcell.gif
Apr 23 2004 10:21:04 208127 ma. -rwxr-xr-x root root /mnt/floppy/Hydrocarbon%20fuel%20cell%20page2.jpg
Apr 23 2004 10:23:24 28167 ma. -rwxr-xr-x root root /mnt/floppy/PEM-fuel-cell-large.jpg
Apr 23 2004 11:21:08 184320 ma. -rwxr-xr-x root root /mnt/floppy/cat.mdb
Apr 23 2004 11:54:32 30720 ma. -rwxr-xr-x root root /mnt/floppy/Remote_Access_Policy.doc
Apr 23 2004 11:55:26 39936 ma. -rwxr-xr-x root root /mnt/floppy/Password_Policy.doc
Apr 23 2004 14:03:54 312 ma. -rwxr-xr-x root root /mnt/floppy/Opportunity.txt
Nov 04 2004 13:55:06 30720 ..c -rwxr-xr-x root root /mnt/floppy/Remote_Access_Policy.doc
Nov 04 2004 13:55:08 184320 ..c -rwxr-xr-x root root /mnt/floppy/cat.mdb
Nov 04 2004 13:55:26 30264 ..c -rwxr-xr-x root root /mnt/floppy/pem_fuelcell.gif
Nov 04 2004 13:55:28 28167 ..c -rwxr-xr-x root root /mnt/floppy/PEM-fuel-cell-large.jpg
Nov 04 2004 13:55:32 208127 ..c -rwxr-xr-x root root /mnt/floppy/Hydrocarbon%20fuel%20cell%20page2.jpg
Nov 04 2004 13:55:42 39936 ..c -rwxr-xr-x root root /mnt/floppy/Password_Policy.doc
Nov 04 2004 13:55:54 32256 ..c -rwxr-xr-x root root /mnt/floppy/Internal_Lab_Security_Policy.doc
Nov 04 2004 13:55:58 312 ..c -rwxr-xr-x root root /mnt/floppy/Opportunity.txt
```

Now, I'd like to add this information to the original timeline. If I remove all entries from Nov. 4 that aren't relevant, the new completed timeline would look as follows.

Sat Feb 03 2001 19:44:16	36864 m.. -/-rwxxrwxrwx 0	0	5	/CamShell.dll (_AMSHLL.DLL) (deleted)
	36864 m.. -rwxxrwxrwx 0	0	5	<fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
Thu Nov 08 2001 21:13:24	194048 m.. -rwxxrwxrwx 0	0	11	<crackedfiles.dd-_AM_PA~1.EXE-dead-11>
	194048 m.. -/-rwxxrwxrwx 0	0	44	/_AM_PA~1.EXE (deleted)
	194048 m.. -/-rwxxrwxrwx 0	0	11	/_AM_PA~1.EXE (deleted)
	194048 m.. -rwxxrwxrwx 0	0	44	<crackedfiles.dd-_AM_PA~1.EXE-dead-44>
Thu Apr 22 2004 16:31:06	32256 m.. -/-rwxxrwxrwx 0	0	75	/Internal_Lab_Security_Policy.doc (INTERN~1.DOC)
	33423 m.. -/-rwxxrwxrwx 0	0	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
	32256 m.. -/-rwxxrwxrwx 0	0	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Fri Apr 23 2004 10:15:18	30264 m.. -/-rwxxrwxrwx 0	0	61	/pem_fuelcell.gif (PEM_FU~1.GIF)
Fri Apr 23 2004 10:21:04	208127 m.. -/-rwxxrwxrwx 0	0	68	/Hydrocarbon_fuel_cell_page2.jpg (HYDROC~1.JPG)
Fri Apr 23 2004 10:23:24	28167 m.. -/-rwxxrwxrwx 0	0	64	/PEM-fuel-cell-large.jpg (PEM-FU~1.JPG)
Fri Apr 23 2004 10:53:56	727 m.. -/-rwxxrwxrwx 0	0	28	/_ndex.htm (deleted)
	727 m.. -rwxxrwxrwx 0	0	28	<fl-260404-RJL1.img-_ndex.htm-dead-28>
Fri Apr 23 2004 11:21:08	184320 m.. -/-rwxxrwxrwx 0	0	58	/CAT.mdb
Fri Apr 23 2004 11:54:32	215895 m.. -/-rwxxrwxrwx 0	0	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
	215895 m.. -/-rwxxrwxrwx 0	0	57	/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26	307935 m.. -/-rwxxrwxrwx 0	0	20	/Password_Policy.doc (PASSWO~1.DOC)
	307935 m.. -/-rwxxrwxrwx 0	0	71	/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:03:54	312 m.. -/-rwxxrwxrwx 0	0	78	/Opportunity.txt (OPPORT~1.TXT)
Fri Apr 23 2004 14:10:50	22528 m.. -/-rwxxrwxrwx 0	0	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10	42496 m.. -/-rwxxrwxrwx 0	0	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Sun Apr 25 2004 00:00:00	0 .a. -/-rwxxrwxrwx 0	0	3	/RJL (Volume Label Entry)
Sun Apr 25 2004 10:53:40	0 m.c -/-rwxxrwxrwx 0	0	3	/RJL (Volume Label Entry)
Mon Apr 26 2004 00:00:00	42496 .a. -/-rwxxrwxrwx 0	0	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
	33423 .a. -/-rwxxrwxrwx 0	0	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
	22528 .a. -/-rwxxrwxrwx 0	0	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
	36864 .a. -/-rwxxrwxrwx 0	0	5	/CamShell.dll (_AMSHLL.DLL) (deleted)
	32256 .a. -/-rwxxrwxrwx 0	0	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
	727 .a. -/-rwxxrwxrwx 0	0	28	/_ndex.htm (deleted)
	36864 .a. -rwxxrwxrwx 0	0	5	<fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
	215895 .a. -/-rwxxrwxrwx 0	0	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
	727 .a. -rwxxrwxrwx 0	0	28	<fl-260404-RJL1.img-_ndex.htm-dead-28>
	307935 .a. -/-rwxxrwxrwx 0	0	20	/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:18	36864 .c -/-rwxxrwxrwx 0	0	5	/CamShell.dll (_AMSHLL.DLL) (deleted)
	36864 .c -rwxxrwxrwx 0	0	5	<fl-260404-RJL1.img-_AMSHLL.DLL-dead-5>
Mon Apr 26 2004 09:46:20	42496 .c -/-rwxxrwxrwx 0	0	9	/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22	32256 .c -/-rwxxrwxrwx 0	0	13	/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24	33423 .c -/-rwxxrwxrwx 0	0	17	/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26	307935 .c -/-rwxxrwxrwx 0	0	20	/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36	215895 .c -/-rwxxrwxrwx 0	0	23	/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44	22528 .c -/-rwxxrwxrwx 0	0	27	/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36	727 .c -/-rwxxrwxrwx 0	0	28	/_ndex.htm (deleted)
	727 .c -rwxxrwxrwx 0	0	28	<fl-260404-RJL1.img-_ndex.htm-dead-28>

Final Completed Timeline with Camouflaged Files Included

I seem to have located what was indicated in Robert's text message. But I want to make sure there aren't hidden files within the hidden files. I decide to again look for 20 00 in any of those extracted files. While I find a number of occurrences, I suspect them to be false positives because when I tried to rerun the camouflage_password_finder.exe against these files, they returned characters outside the standard ASCII set. See below as an example of the password cracker's output.



I examined the contents of each of these hidden files and have them attached here. It would appear that these are the files containing the confidential information and what Robert had intended to smuggle out.

Clients : Table											
	First	Last	Phone	Company	Address	Address1	City	State	Zipcode	Account	Password
▶	Bob	Esposito	703-233-2048	Cook Labs	245 Main St		Alexandria	VA	20231	espomain	y4NSHMNF
	Jerry	Jackson	410-677-7223	Double J's	11561 W. 27 St.		Baltimore	MD	20278	jack27st	JLbW3Pg5
	David	Lee	866-554-0922	Tech Vision	300 Lone Grove Lane		Wichita	KS	30189	leetechn	O1A26a3k
	Marie	Horton	800-234-king	King Labs, Inc.	700 King Labs Ave	Suite 900	Biloxi	MS	39533	hortking	Yk7Sr4pA
	Lenny	Jones	877-Get-done	Quick Printing	99 E. Grand View Dr		Omaha	NE	56098	joneeast	868y48RH
	Jeff	Hayes	404-893-5521	Big Sky First	90 Old Saw Mill Rd		Billings	MT	59332	hayeolds	3R30bb7i
	Roger	Forrester	210-586-2312	TCFL	188 Greenville Rd		Austin	TX	77239	forrgree	si4OW8UV
	Edward	Cash	212-562-0997	E & C Inc.	76 S. King St	Suite 300	Santa Barbara	CA	80124	cashking	O8uQ1fC
	Steve	Bei	616-833-0129	Island Labs	65 Kiwi Way		Honolulu	HA	93991	beikiwiw	JDH20u26
	Jodie	Kelly		Data Movers	7256 Beerwah Ave.	Suite 110	Wetherby	U.K.	LS22 6RG	kellbeer	tmu0ENOk
	Patrick	Roy		The Magic Lam	4150 Regents Park	Row #170	Calgary	CAN	R4316DF	roythema	rJag6Q0O

CAT.mdb

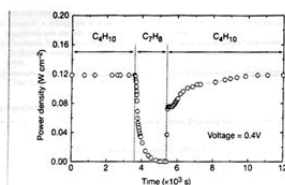


Figure 3 Effect of switching fuel type on the cell with the Cu-ceria composite anode at 973 K. The power density of the cell is shown as a function of time. The fuel was switched from *n*-butane (C_4H_{10}) to toluene (C_7H_8) and back to *n*-butane.

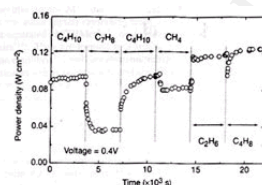
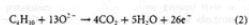


Figure 4 Effect of switching fuel type on the cell with the Cu-doped ceria composite anode at 973 K. The power density is shown as a function of time. The fuels were: *n*-butane (C_4H_{10}), toluene (C_7H_8), *n*-butane, methane (CH_4), ethane (C_2H_6), and 1-butene (C_4H_8).

higher temperature. Visual inspection of a cell after two days in *n*-butane at 1,073 K showed that the anode itself remained free of the tar deposits that covered the alumina walls.

Although it is possible that the power generated from *n*-butane fuels resulted from oxidation of H_2 —formed by gas-phase reactions of *n*-butane that produce hydrocarbons with a lower C:H ratio—other evidence shows that this is not the case. First, experiments were conducted in which the cell was charged with *n*-butane and then operated in a batch mode without flow. After 30 minutes of batch operation with the cell short-circuited, GC analysis showed that all of the *n*-butane in the cell had been converted completely to CO_2 and water. (Negligible amounts of CO_2 were formed in a similar experiment with an open circuit.) Second, analysis of the CO_2 formed under steady-state flow conditions, shown in Fig. 2, demonstrates that the rate of CO_2 formation increased linearly with the current density. (It was not possible for us to quantify the amount of water formed in our system.) Figure 2 includes data for both *n*-butane at 973 K, and methane at 973 K and 1,073 K. The lines in the figure were calculated assuming complete oxidation of methane (the dashed line) and *n*-butane (the solid line) to CO_2 and water according to reactions (1) and (2):



With methane, only trace levels of CO were observed along with CO_2 , so that the agreement between the data points and the calculation demonstrates consistency in the measurements and no leaks in the cell. With *n*-butane, simultaneous, gas-phase, free-radical reactions to give hydrocarbons with various C:H ratios make quantification more difficult; however, the data still suggest that complete oxidation is the primary reaction. Furthermore, the batch experiments show that the secondary products formed by gas-phase reactions are ultimately oxidized as well. Taken together, these results demonstrate the direct, electrocatalytic oxidation of a higher hydrocarbon in a SOFC.

Along with our observation of stable power generation with *n*-butane for 48 hours, Fig. 3 further demonstrates the stability of the composite anodes against coke formation. Aromatic molecules, such as toluene, are expected to be precursors to the formation of graphitic coke deposits. In Fig. 3, the power density was measured at 973 K and 0.4 V while the fuel was switched from dry *n*-butane, to 0.033 bar of toluene in He for 30 minutes, and back to dry *n*-butane. The data show that the performance decreased rapidly in the presence of toluene. Upon switching back to dry *n*-butane, however,

the current density returned to 0.12 $W\ cm^{-2}$ after one hour. Because the return was not instantaneous, it appears that carbon formation occurred during exposure to toluene, but that the anode is self-cleaning. We note that the electrochemical oxidation of soot has been reported by others¹¹.

The data in Fig. 4 show that further improvements in cell performance can be achieved. For these experiments, samaria-doped ceria was substituted for ceria in the anode, and the current densities were measured at a potential of 0.4 V at 973 K. The power densities for H_2 and *n*-butane in this particular cell were approximately 20% lower than for the first cell, which is within the range of our ability to reproduce cells. However, the power densities achieved for some other fuels were significantly higher. In particular, stable power generation was now observed for toluene. Similarly, Fig. 4 shows that methane, ethane and 1-butene could be used as fuels to produce electrical energy. The data show transients for some of the fuels, which are at least partially due to switching.

The role of samaria in enhancing the results for toluene and some of the other hydrocarbons is uncertain. While samaria is used to enhance mixed (ionic and electronic) conductivity in ceria and could increase the active, three-phase boundary in the anode, samaria is also an active catalyst¹². Other improvements in the performance of SOFCs are possible. For example, the composite anodes could be easily attached to the cathode-supported, thin-film electrolytes that have been used by others to achieve very high power densities¹. In addition to raising the power density, thinner electrolytes may also allow lower operating temperatures.

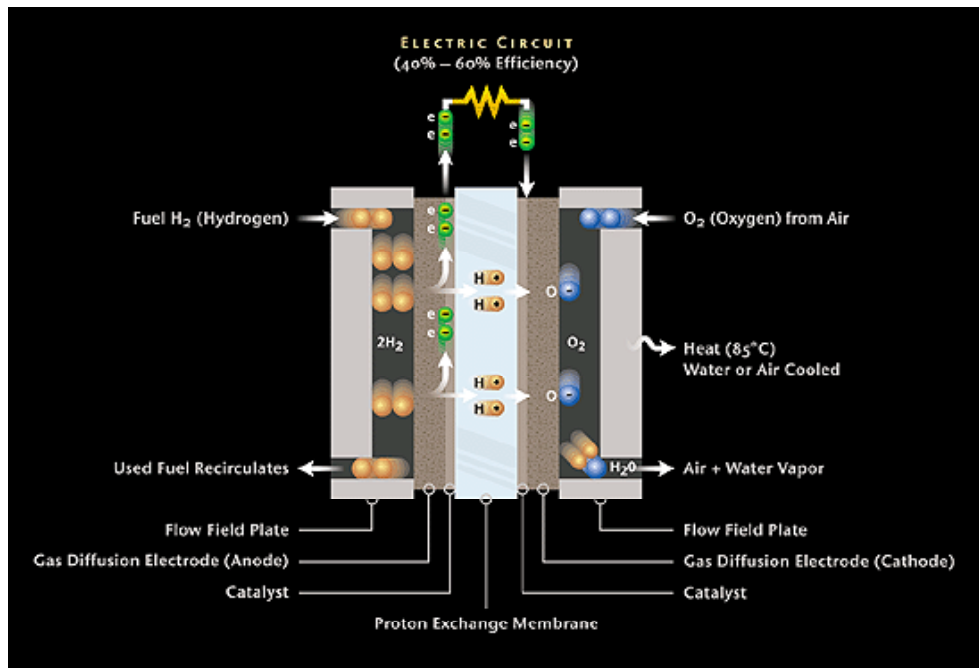
Additional research is clearly necessary for commercial development of fuel cells which generate electrical power directly from hydrocarbons; however, the work described here suggests that SOFCs have an intriguing future as portable, electric generators and possibly even as energy sources for transportation. The simplicity afforded by not having to reform the hydrocarbon fuels is a significant advantage of these cells.

Received 13 September 1999; accepted 20 January 2000.

1. Steele, B. C. H. Running on natural gas. *Nature* **400**, 620–621 (1999).
2. Serizawa, E. E. Bringing fuel cells down to earth. *Science* **285**, 482–485 (1999).
3. Perry, M. J., E. T. & B. B. A. A direct-methane fuel cell with a ceria-based anode. *Nature* **400**, 440–441 (1999).
4. Pinnau, I. S., S. Subramanian, J. V. & M. B. G. R. I. Ceria-based anodes for the direct oxidation of methane in solid oxide fuel cells. *Langmuir* **15**, 4832–4837 (1999).
5. Pinnau, I. S., S. Subramanian, J. V. & M. B. G. R. I. Direct oxidation of hydrocarbons in a solid oxide fuel cell: I. methane oxidation. *J. Electrochem. Soc.* **146**, 3603–3605 (1999).
6. Steele, B. C. H., S. Subramanian, J. V. & M. B. G. R. I. Oxidation of methane in solid-state electrochemical reactors. *Solid State Ionics* **28**, 1167–1172 (1988).
7. Lloyd, A. C. The power plant in your basement. *Sci. Am.* **281**(12), 80–86 (1999).

Supplementary analysis available online.

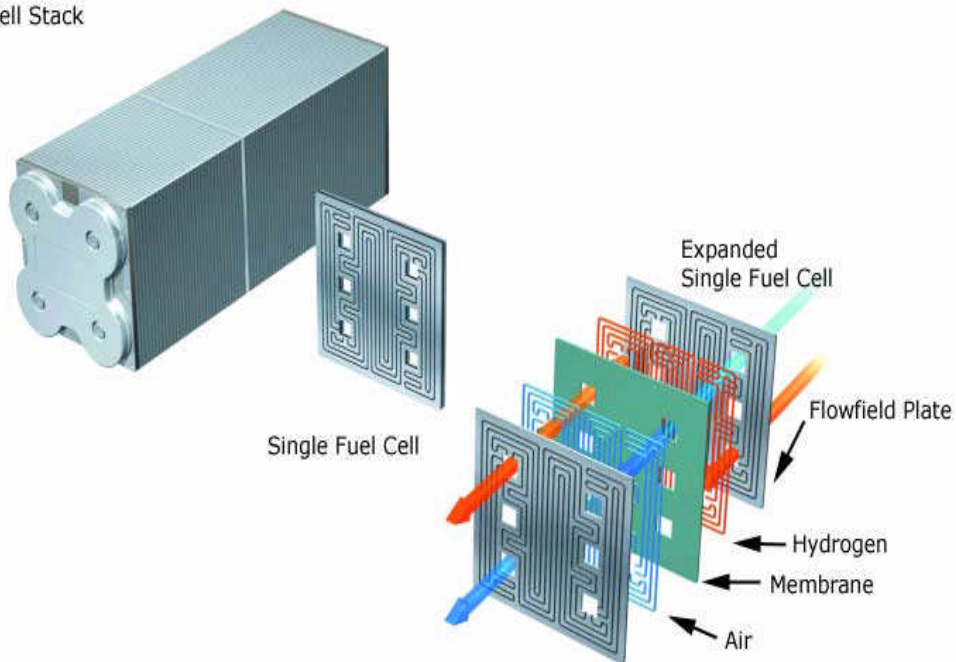
Hydrocarbon%20fuel%20cell%20page2.jpg



pem_fuelcell.gif

Design of a PEM Fuel Cell

Fuel Cell Stack



PEM-fuel-cell-large.jpg

As far as file analysis goes, there's still one more thing that I need to understand, and that is the significance of the deleted file, *_ndex.htm* and why that same HTML code shows up in *CamShell.dll*.

You'll recall that I'd already extracted the contents of the file located at inode 5 earlier, called "*_AMSHHELL.DLL.inode5*". At this point, I search the internet with Google to locate any/all versions of the Camouflage program. I locate 3 versions.

Name	Size	Type	Date Modified
camou104.zip	1,367 KB	WinZip File	11/12/2004 2:53 PM
camou111.exe	2,653 KB	Application	11/12/2004 3:14 PM
Camou121.exe	2,655 KB	Application	11/3/2004 4:23 PM

I transfer all three versions to my test, non-networked Windows XP box and, one at a time, install each of the versions. Each version has its own version of the *camshell.dll* (except for *Camou121.exe*, which had *Camouflage.dll* and was the first version I had used to extract the password locked files). I open each *.dll* within a hex editor and scan the file looking for a section that speaks towards the file version. The *dll* from *camou111.exe* seems to match file version from the *camshell.dll.inode5*. I had obtained this copy of the file from <http://files.letoltes.com/biztonsag/camou111.exe>. It's also important to note that, when I installed *camou111.exe*, the install splash screen proclaimed this was version 1.1.2. I would have expected it to say it was version 1.1.1 judging from the file name, but I chalk this up as a minor mystery and proceed with the analysis, hereafter referring to the extracted *dll* as "*CamShell.dll.112*".

I then what to explore the difference between "*_AMSHHELL.DLL.inode5*" and "*CamShell.dll.112*". I first dump each file to hex with ASCII translation, and then run the linux "*diff*" command against each the two resultant hexdumped files. The results are as follows.

```
[root@LinuxForensics giac]# hexdump -v -C _AMSHHELL.DLL.inode5 >
_AMSHHELL.DLL.inode5.hexdump
[root@LinuxForensics giac]# hexdump -v -C CamShell.dll.112 > CamShell.dll.112.hexdump
[root@LinuxForensics giac]# diff _AMSHHELL.DLL.inode5.hexdump CamShell.dll.112.hexdump
1,46c1,46
< 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 |
< 000001a0 4e 41 4d 45 3d 71 75 61 6c 69 74 79 20 56 41 4c |NAME=quality VAL|
< 000001b0 55 45 3d 68 69 67 68 3e 20 3c 50 41 52 41 4d 20 |UE=high> <PARAM |
< 000001c0 4e 41 4d 45 3d 62 67 63 6f 6c 6f 72 20 56 41 4c |NAME=bgcolor VAL|
< 000001d0 55 45 3d 23 43 43 43 43 43 43 3e 20 3c 45 4d 42 |UE=#CCCCC> <EMB|
< 000001e0 45 44 20 73 72 63 3d 22 62 61 6c 6c 61 72 64 2e |ED src="ballard.|
< 000001f0 73 77 66 22 20 71 75 61 6c 69 74 79 3d 68 69 67 |swf" quality=hig|
< 00000200 68 20 62 67 63 6f 6c 6f 72 3d 23 43 43 43 43 43 |h bgcolor=#CCCCC|
< 00000210 43 20 20 57 49 44 54 48 3d 22 38 30 30 22 20 48 |C WIDTH="800" H|
< 00000220 45 49 47 48 54 3d 22 36 30 30 22 20 4e 41 4d 45 |EIGHT="600" NAME|
< 00000230 3d 22 62 61 6c 6c 61 72 64 22 20 41 4c 49 47 4e |= "ballard" ALIGN|
< 00000240 3d 22 22 0d 0a 20 54 59 50 45 3d 22 61 70 70 6c |= "... TYPE="appl|
< 00000250 69 63 61 74 69 6f 6e 2f 78 2d 73 68 6f 63 6b 77 |ication/x-shockw|
< 00000260 61 76 65 2d 66 6c 61 73 68 22 20 50 4c 55 47 49 |ave-flash" PLUGI|
< 00000270 4e 53 50 41 47 45 3d 22 68 74 74 70 3a 2f 2f 77 |NSPAGE="http://w|
< 00000280 77 77 2e 6d 61 63 72 6f 6d 65 64 69 61 2e 63 6f |ww.macromedia.co|
< 00000290 6d 2f 67 6f 2f 67 65 74 66 6c 61 73 68 70 6c 61 |m/go/getflashpla|
< 000002a0 79 65 72 22 3e 3c 2f 45 4d 42 45 44 3e 0d 0a 3c |yer"></EMBED>...<|
< 000002b0 2f 4f 42 4a 45 43 54 3e 0d 0a 3c 2f 63 65 6e 74 |/OBJECT>...</cent|
< 000002c0 65 72 3e 0d 0a 3c 2f 42 4f 44 59 3e 0d 0a 3c 2f |er>...</BODY>...</|
< 000002d0 48 54 4d 4c 3e 0d 0a 00 00 00 00 00 00 00 00 |HTML>.....|

```

© SANS Institute 2005

I've added some coloring to make it easier to read. The content in gray shows the portion of the `_AMSHHELL.DLL.inode5` that differs from the content in `CamShell.dll.112`. Note that the HTML content seems to be the only difference between the files, that beginning at byte offset `0x02e0`, the files are identical.

I decide to rerun the `hexdump` command, this time extracting all contents after `0x02e0` from each file and running `md5sums` against each. They match identically.

```
[root@LinuxForensics giac]# hexdump -s 0x2e0 -C -v _AMSHHELL.DLL.inode5 > test.txt
[root@LinuxForensics giac]# hexdump -s 0x2e0 -C -v CamShell.dll.112 > test2.txt
[root@LinuxForensics giac]# md5sum test.txt test2.txt
57446b339bb23bbad76a64cd5906bbc6 test.txt
57446b339bb23bbad76a64cd5906bbc6 test2.txt
[root@LinuxForensics giac]#
```

I feel comfortable concluding at this point that the version of the program he was using was Camouflage version 1.1.2.

I theorize at this point that the deleted files `_ndex.htm` and `_amshell.dll` were early attempts by Robert using steganography. He looks like he's experimenting with embedding files within files, perhaps before he'd mastered camouflage. Perhaps he added the HTML content to the `_AMSHHELL.DLL.inode5` file in an attempt to hide the DLL. Perhaps this is why the files show as deleted, because they were early, failed attempts. The `index.htm` looks like it might have come from a legitimate Ballard Industries website.

File Summary

During the preparation of this practical, it became readily apparent to me that one could get easily lost in all the similar sounding file names and what they all are supposed to be. I decided to create this handy index of some of the more important file names, file descriptions, and `md5sums` for anyone wanting to

reference it.

<u>File Name</u>	<u>File Description</u>	<u>MD5</u>
_AMSHLL.DLL.inode5	Exported from Autopsy using sorter, extract of CamShell.dll.	6462fb3acca0301e52fc4ffa4ea5efff8
_ndex.htm	Deleted File within fl-260404-RJL1.img, located at inode 28	17282ea308940c530a86d07215473c79
Acceptable_Encryption_Policy.doc	File within fl-260404-RJL1.img, located at inode 27	f785ba1d99888e68f45dabeddb0b4541
camoul04.zip	Install file for Camouflage Version 1.04	42d78d90f7c4eda3317f3a4504fb031c
camoul11.exe	Install file for Camouflage Version 1.11 (Setup splash screen claims to be version 1.1.2)	6d9f47e69716230ceb0fbdab30694737
camoul21.exe	Install file for Camouflage Version 1.21	c62b050117c2cba3518e5a734fedef1f
CamouflageShell.dll	File from camoul21.exe	cb4e5354a056563ddb39b4d79f0c7dc9
CamShell.dll	Deleted File within fl-260404-RJL1.img, located at inode 5	6462fb3acca0301e52fc4ffa4ea5efff8
CamShell.dll.112	Originally called CamShell.dll from file within setup for camoul11.exe, renamed by Mike Aylor	4e986ab0909d2946bed868b5f896906f
cat.mdb	File hidden within Remote_Access_Policy.doc	c3a869ff6b71c7be3eb06b6635c864b1
fl-260404-RJL1.img	File presented to us by David Keen containing all known forensic evidence	d7641eb4da871d980adbe4d371eda2ad
fl-260404-RJL1.img.dls	Contains unallocated data from the fl-260404-RJL1.img file. Generated by Autopsy	n/a
fl-260404-RJL1.img.dls.str	Contains strings obtained from fl-260404-RJL1.img.dls. Generated by Autopsy	n/a
fl-260404-RJL1.img.str	Contains strings from the allocated portion of fl-260404-RJL1.img. Generated by Autopsy.	n/a
Hydrocarbon%20fuel%20cell%20page2.jpg	File hidden within Password_Policy.doc (md5: ac3...).	9da5d4c42fdf7a979ef5f09d33c0a444
Information_Sensitivity_Policy.doc	File within fl-260404-RJL1.img, located at inode 9	99c5dec518b142bd945e8d7d2fad2004
Internal_Lab_Security_Policy.doc	File within fl-260404-RJL1.img, located at inode 17	b9387272b11aea86b60a487fbdc1b336
Internal_Lab_Security_Policy.doc	File hidden within Internal_Lab_Security_Policy.doc (md5: b93...)	e0c43ef38884662f5f27d93098e1c607
Internal_Lab_Security_Policy1.doc	File within fl-260404-RJL1.img, located at inode 13	e0c43ef38884662f5f27d93098e1c607
Opportunity.txt	File hidden within Internal_Lab_Security_Policy.doc (md5: b93...)	3ebd8382a19c88c1d276645035e97ce9

Password_Policy.doc	File within fl-260404-RJL1.img, located at inode 20	ac34c6177ebdcfaf4 adc41f0e181belbc
Password_Policy.doc	File hidden within Password_Policy.doc (md5: ac3...).	e5066b0fb7b91add 563a400f042766e4
pem_fuelcell.gif	File hidden within Password_Policy.doc (md5: ac3...).	864e397c2f38ccfb 778f348817f98b91
PEM-fuel-cell-large.jpg	File hidden within Password_Policy.doc (md5: ac3...).	5e39dcc44acccdca 7bba0c15c6901c43
Remote_Access_Policy.doc	File within fl-260404-RJL1.img, located at inode 23	5b38d1ac1f94285d b2d2246d28fd07e8
Remote_Access_Policy.doc	File hidden within Remote_Access_Policy.doc	2afb005271a93d44 b6a8489dc4635c1c

Legal Implications

There is considerable, but not conclusive, evidence that Robert J. Lesczynski attempted to sell company trade secrets to competitors. First, there is evidence that company trade secrets were present on the seized floppy disk in the form of a files; a customer database and technical documents related to proprietary fuel cells. There is evidence that the files were modified so as to attempt to hide their presence on the floppy disk, in the form of the program “Camouflage”. The file named “Opportunity.txt” indicates Robert’s intention to sell the files for profit to a competitor. Tying this evidence specifically too Robert is more problematic, since none of the information gleaned from the floppy demonstrates that Robert, and only Robert, was the one who put the proprietary information on the floppies. There is no evidence of his username or other identifying credentials present on the floppy. It’s true that the “Opportunity.txt” contains his name, however, this is only circumstantial since his name could easily have been forged in the text file. So far, the only good evidence linking Robert to the incident is he had possession of the floppy when the security guard seized it.

For further evidence, I would recommend conducting an investigation of any workstation Robert may have had access to at work, looking for the presence of the Camouflage program and the proprietary files in question. I would recommend examining any records showing what websites he’d visited and what programs he may have downloaded from the Internet, if any such logs exist. If network logins or physical door access (door badges, cameras, etc.) are recorded, I would try to correlate those to the time stamps present on the floppy, i.e., look for his activity around 9:46am 4/26/2004.

There are applicable United States statutes for this situation, specifically 18 U.S.C. 1831 and 1832. Excerpts are provided below...

§ 1831. Economic espionage

(a) In General.— Whoever, intending or knowing that the offense will benefit any foreign government, foreign instrumentality, or foreign agent, knowingly—

(1) steals, or without authorization appropriates, takes, carries away, or conceals, or by fraud, artifice, or deception obtains a trade secret;

(2) without authorization copies, duplicates, sketches, draws, photographs, downloads, uploads, alters, destroys, photocopies, replicates, transmits, delivers, sends, mails, communicates, or conveys a trade secret;

(3) receives, buys, or possesses a trade secret, knowing the same to have been stolen or appropriated, obtained, or converted without authorization;

(4) attempts to commit any offense described in any of paragraphs (1) through (3); or

(5) conspires with one or more other persons to commit any offense described in any of paragraphs (1) through (3), and one or more of such persons do any act to effect the object of the conspiracy,

shall, except as provided in subsection (b), be fined not more than \$500,000 or imprisoned not more than 15 years, or both.

(b) Organizations.— Any organization that commits any offense described in subsection (a) shall be fined not more than \$10,000,000.

§ 1832. Theft of trade secrets

(a) Whoever, with intent to convert a trade secret, that is related to or included in a product that is produced for or placed in interstate or foreign commerce, to the economic benefit of anyone other than the owner thereof, and intending or knowing that the offense will, injure any owner of that trade secret, knowingly—

(1) steals, or without authorization appropriates, takes, carries away, or conceals, or by fraud, artifice, or deception obtains such information;

(2) without authorization copies, duplicates, sketches, draws, photographs, downloads, uploads, alters, destroys, photocopies, replicates, transmits, delivers, sends, mails, communicates, or conveys such information;

(3) receives, buys, or possesses such information, knowing the same to have been stolen or appropriated, obtained, or converted without authorization;

(4) attempts to commit any offense described in paragraphs (1) through (3); or

(5) conspires with one or more other persons to commit any offense described in paragraphs (1) through (3), and one or more of such persons do any act to effect the object of the conspiracy, shall, except as provided in subsection (b), be fined under this title or imprisoned not more than 10 years, or both.
(b) Any organization that commits any offense described in subsection (a) shall be fined not more than \$5,000,000.

Thus, action could likely be pursued for corporate espionage and theft of trade secrets.

If the corporate policy documents found on the floppy image are current, then there is evidence of company policy violations as well. Specifically, the "Information Sensitivity Policy" describes various categories of sensitive information, ranging from Minimal Sensitivity, More Sensitive, and Most Sensitive. It describes Most Sensitive as "Trade secrets & marketing, operational, personal, financial, source code, & technical information integral to the success of our company." The aforementioned hidden files would appear to fall into this category.

For most sensitive information, only non-employees who are designated with approved access and signed non-disclosure agreements are eligible. In addition, distribution of said information outside of company internal mail should be handled by delivered direct mail, with signature required and by approved private carriers. Furthermore, electronic distribution is highly recommended to be strongly encrypted. None of the above guidelines were followed.

Ironically, the company's password policy was also violated. The sensitive files were protected using a simple password scheme, those being "Password" and "Remote". One file was protected with only a blank password. Per company policy, all passwords should be strong passwords, defined as:

More than eight characters ("Remote" has six).
Are not a word in any language (both passwords are)

Furthermore, a hint was given as to the syntax of the password within the "Opportunity.txt" file. Specifically, Robert indicates "*They are available as we discussed - 'First Name'*". "Password" is the first name off of the file name "Password_Policy.doc" and "Remote" the first from "Remote_Access_Policy.doc". However, the policy expressly forbids "hinting at the format of a password".

Lastly, the company's Acceptable Encryption Policy was violated, when the sensitive files were stored not using a proven strong algorithm. The algorithm used, in fact, was proprietary and, according to the policy, "The use of

proprietary encryption algorithms is not allowed for any purpose, unless reviewed by qualified experts outside of the vendor in question and approved by InfoSec.” It may be necessary to investigate if InfoSec ever approved the usage of a tool such as Camouflage.

Part II – Perform Forensic Tool Validation.

For this section, I have chosen to perform analysis on X-Ways Forensics 12.0 SR-13, by X-Ways Software Technology AG and authored by Stefan Fleischman. See <http://www.x-ways.com/> for additional information.

The tool is, at its core, a hex editor capable of examining files and disks at the bit/byte level, but unlike most basic hex editors, this tool is geared towards computer forensic investigations. It has the ability to read image files generated by dd or EnCase and interpret them as an actual file system, thus allowing the analyst to extract deleted files, search for all types of files that might be hidden, clone disk images, show MAC date/time stamps in calendar format, etc. It has the ability to perform case management, allowing the analyst to append notes and excerpts as they conduct their investigation. Furthermore, it has been toolled specifically for forensic investigators – unless explicitly told to, it will not allow even accidental modification of opened images, thus preserving their forensic integrity (X-Ways Forensics Software AG, 2004).

For the purpose of this paper, I will focus exclusively on the tool’s ability to perform disk cloning. The tool literature indicates that it performs disk cloning in a forensically sound fashion. Ideally, a cloned image is identical in every respect to the original source, which should be determined by performing an MD5 sum of both the source and the cloned image. They should match exactly.

Installation of the tool requires a PC running some variant of Microsoft Windows (9x/ME/NT/2000/XP). Full details of setup considerations can be found at <http://www.x-ways.net/winhex/setup.html>. The tool is also able to be loaded directly from a bootable CD using Bart’s PE Builder. The tool is commercial software and its source code wasn’t immediately apparent on any of the sponsoring X-Ways websites, and it is installed using a setup utility, so there is no compiling needed. Normally, the tool requires a license activation to leverage some of its unique functionality. I was able to arrange the legitimate purchase of a licensed copy from X-Ways.

Test Environment

1. Windows Machine – Dell Optiplex GX270 2.6 GHZ running Windows XP SP 1 with 1 GB RAM. Has X-Ways Forensics 12.0 SR-13 installed, and HashCalc 2.01 installed.
2. Linux Machine – Dell Optiplex GX260 1.0 GHZ running RedHat Fedora Core 2 with 512 MB RAM.

Both systems are isolated from any network, including themselves. They are located on my desk at work in a physically segregated section of the floor.

Testing Procedures

Hypothesis: According to X-Ways website, X-Ways Forensics is capable of performing “Disk cloning and imaging, even under DOS with X-Ways Replica (forensically sound)”. A forensically sound image is one that has not altered even in the slightest way from the original source. If the tool performs as advertised, there should be no detectable difference between any source and any X-Way’s generated clone. The focus of the tests will be on X-Ways Forensics, and not the X-Ways Replica tool for DOS. In addition, X-Ways has the ability to compute the MD5 and SHA-1 hashes (among other types of hashes) of images and disks. If the tool is written properly, hashes generated by the tool should be identical to hashes generated by any other non-X-Ways, 3rd party tool.

Step 1. Baselineing

In order to determine the authenticity of reproduced disk images, we will use two common algorithms; MD5 and SHA-1 against the image files and media. It is important to use both algorithms because MD5 was recently shown to possibly contain flaws in the manner it calculates hashes. Precisely how those flaws could be exploited warrants another practical paper all its own, but suffice it to say that it is mathematically possible to generate “two identical outputs from a hash function” (<http://www.internetnews.com/security/article.php/3446071>).

We will use the “md5sum” and “sha1sum” binaries that come with the RedHat Fedora Core 2 install, and HashCalc (described in Part 1) on Windows. By using two separate tools, two separate platforms, and two separate algorithms, we eliminate any reasonable doubt that the tools are trustworthy.

To establish a baseline, we will first obtain a copy of the floppy image used in Part I of this practical. Since the MD5 hash was indicated within the practical assignment description, the first step will be to examine this image file with both HashCalc and md5sum and ensure both match. Afterwards, determine what

the SHA-1 hashes for the image file are by using both HashCalc and sha1sum. Again, look for a match. Since we haven't been given SHA1 hashes within the practical, we will be relying exclusively on our tool's ability to match hashes as validation of our test. But, assuming the SHA1 hashes do match, it is then reasonable to assume that neither of the tools has a flaw in the way it computes the hashes (or at least, they are flawed identically, which is very unlikely).

Assuming everything tested here matches, we can proceed forward with reasonable assurance that the tools we are using to compute hashes are reliable.

Step 2. Cloning from the original image file to the X-Ways forensics image file.

A copy of the fl-260404-RJL1.img will be transferred by USB Key Chain drive to the Windows machine. HashCalc will be run to confirm the MD5 and SHA-1 hashes of the image file.

X-Ways Forensics will be started and we will open the fl-260404-RJL1.img file inside it. By default, the tool will open it as a normal file, presenting the file as a raw hex dump. By selecting the option "Interpret Image File as Disk" under the "Specialist" Tab, as the name suggests, it will open the file as a disk, displaying relevant file and directory information.

By selecting Tools -> Disk Tools -> Clone Disk, we will create an X-Ways Forensics generated image file. HashCalc will perform an MD5 and SHA-1 hash calculation on the new file, and we will compare the results to our baseline.

If they do not match, the new file will be transferred via USB Key Chain drive to the Linux machine, rerun the MD5 and SHA-1 calculations, confirm the file transferred correctly, and then run a series of hexdumps and diffs against the two files to determine what changed between the old and new image files.

Step 3. Cloning from the disk media to the X-Ways forensics image file.

Take the original image file and transfer it to a blank floppy disk using Linux dd command. Run MD5 and SHA-1 hashes of the floppy and confirm it matches the original image file. Write protect the floppy by flipping the plastic tab on the corner of the diskette.

By selecting Tools -> Disk Tools -> Clone Disk, and choosing the logical floppy disk option as the source, we will clone from the floppy disk to an X-Ways generated image file. We will then perform MD5 and SHA1 hash calculations using HashCalc, confirm that they match against the original image file.

If they do not match, perform steps to detect differences as outlined in Step 2.

Step 4. Cloning from the original image file to disk media.

Open the original image file in X-Ways Forensics and select Tools -> Disk Tools -> Clone Disk, this time specifying the Logical Disk of the floppy drive as the image destination. Once the cloning is completed, run HashCalc against the diskette and confirm the MD5 and SHA-1 hashes.

Step 5. Confirm the hash calculation function of X-Ways Forensics.

X-Ways Forensics has its own hash algorithm calculator. If we perform hash calculations of the original image file, the X-Ways Forensics generated image file, the floppy disk created by dd, and the floppy disk created by X-Ways Forensics, they should all match, not only with each other but also with those hashes generated by HashCalc, md5sum, and sha1sum.

Nomenclature/Terminology

<u>File Name</u>	<u>File Description</u>
FI-260404-RJL1.img	original image file provided in Part 1 of the practical.
XF1.img	image file generated by X-Ways Forensics (omitted clusters)
XF1a.img	image file generated by X-Ways Forensics (? BAD CLUSTER ?)
XF2.img	image file generated by X-Ways Forensics (NULL, correct hash)
XF3.img	image file generated by X-Ways Forensics (physical disk)
flhexdump.txt	text file containing hexdump information from fl-260404-RJL1.img
xf1.txt	text file containing hexdump information from xf1.img
xf2.txt	text file containing hexdump information from xf2.img

Note: A more thorough discussion of all file names, file descriptions, and hashes will occur later in this practical.

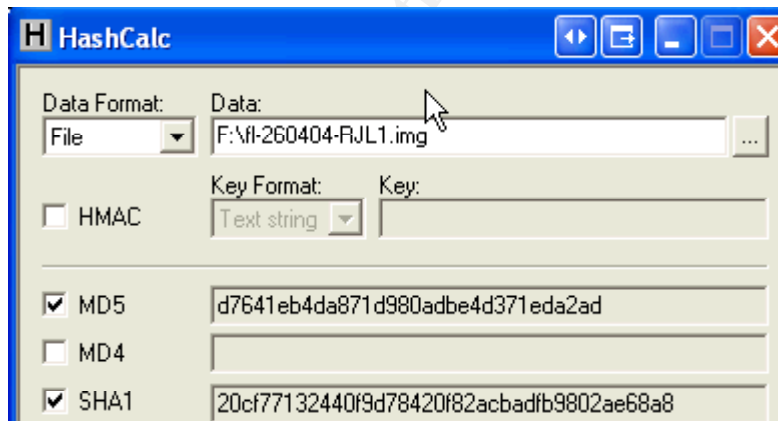
Data Results

Test 1 - Baselineing

The FL-260404-RJL1.img file is copied to the USB keychain drive and mounted on the Linux machine as a read-only file system. I confirm it is read only by attempting to copy the /etc/services file to the USB drive, and it generates an expected error. I then run an md5sum and a sha1sum of the file.

```
[root@LinuxForensics mnt]# mount -o ro /dev/sda1 /mnt/usbdrive
[root@LinuxForensics mnt]# cd /mnt/usbdrive/
[root@LinuxForensics usbdrive]# cp /etc/services /mnt/usbdrive
cp: cannot create regular file `/mnt/usbdrive/services': Read-only file system
[root@LinuxForensics usbdrive]# md5sum fl-260404-RJL1.img
d7641eb4da871d980adbe4d371eda2ad  fl-260404-RJL1.img
[root@LinuxForensics usbdrive]# sha1sum fl-260404-RJL1.img
20cf77132440f9d78420f82acbadfb9802ae68a8  fl-260404-RJL1.img
[root@LinuxForensics usbdrive]#
```

The USB drive is umounted from the Linux machine and plugged into the Windows machine. HashCalc is run on the image file and it shows the MD5 and SHA-1 hashes.



A character by character comparison of the hashes generated on the Linux machine and the Windows machine show that the hashes are identical.

When we compare these results with the MD5 hash given to us by “David Keen”

The security administrator, David Keen, has asked you to analyze the floppy disk and provide a report of your findings prior to returning it to Robert. He provides you with a chain of custody form with the following information:

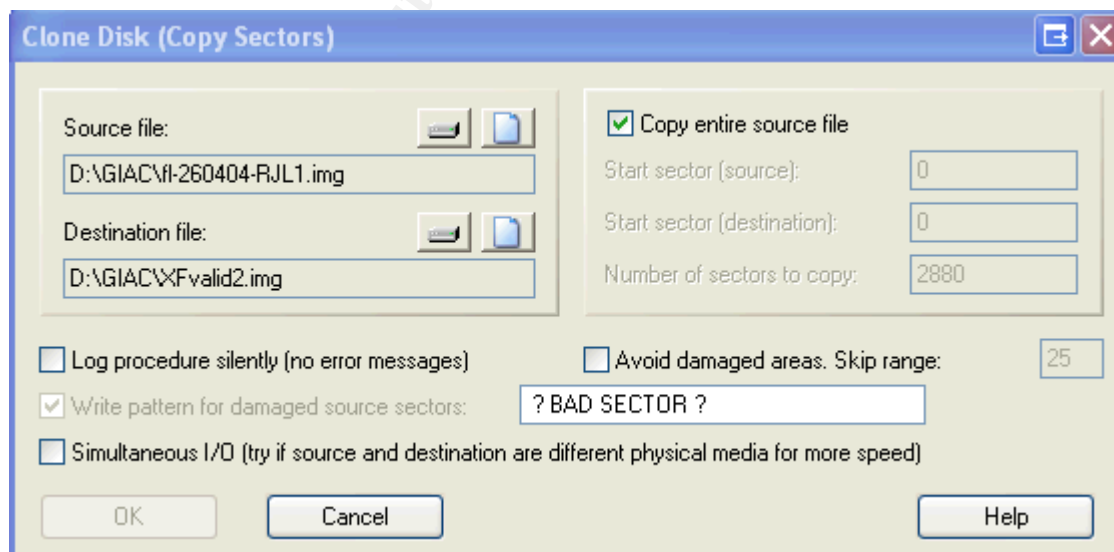
- ◆ Tag# fl-260404-RJL1
- ◆ 3.5 inch TDK floppy disk
- ◆ MD5: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
- ◆ fl-260404-RJL1.img.gz

...we see they match. And so, the baselines for the file for all future tests will be:

MD5: d7641eb4da871d980adbe4d371eda2ad
SHA-1: 20cf77132440f9d78420f82acbadfb9802ae68a8

Test 2 - Cloning from the original image file to the X-Ways forensics image file.

Now, we will determine if cloning from a dd generated image file to an X-Ways Forensic generated image file is forensically sound. We use the fl-260404-RJL1.img file already on the Windows machine that we confirmed the MD5 and SHA-1 hashes for. We then open up the file within X-Ways Forensics and select Tools -> Disk Tools -> Clone Disk. We specify the fl-260404-RJL1.img as the source and XFvalid2.img as the destination.



Note that the "OK" button is grayed out. After much trial and error, it has been determined that it will remain grayed out until either the source or destination is

made to be physical media (i.e. an actual floppy disk). X-Ways Forensics will not allow a copy of an image file to be created from an image file. However, assuming you have enough physical drives, it will allow you to copy from physical media to another physical media (of greater than or equal to size).

Since no data could be gathered for this item, it will be omitted from any future analysis in this practical.

Test 3 - Cloning from the disk media to the X-Ways forensics image file.

First, we will want to wipe any floppy diskette to ensure there are no traces of data on it, even if it came from a fresh unopened box of floppies. To accomplish this, we insert the disk into the Linux machine and use dd to write all zeroes to the floppy disk.

```
[root@LinuxForensics mnt]# dd if=/dev/zero of=/dev/fd0
dd: writing to `/dev/fd0': No space left on device
2881+0 records in
2880+0 records out
```

Once this is completed, we then transfer the original image from the USB keychain drive to the floppy again using dd.

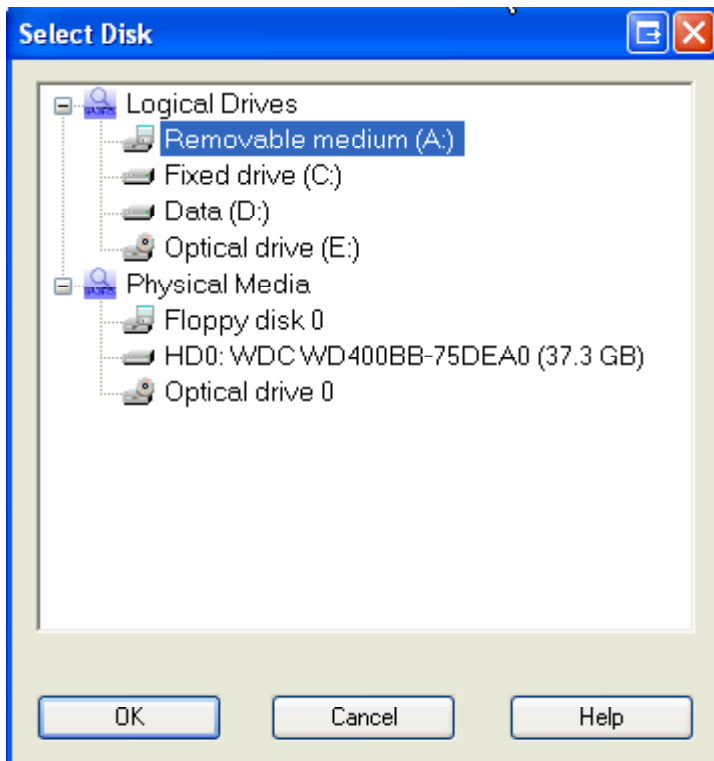
```
[root@LinuxForensics mnt]# dd if=/mnt/usbdrive/fl-260404-RJL1.img of=/dev/fd0
2880+0 records in
2880+0 records out
[root@LinuxForensics mnt]#
```

From the Linux machine, we run MD5 and SHA-1 hashes against the newly created floppy, and compare to the results from our baseline in step 1.

```
[root@LinuxForensics mnt]# md5sum /dev/fd0
d7641eb4da871d980adbe4d371eda2ad /dev/fd0
[root@LinuxForensics mnt]# sha1sum /dev/fd0
20cf77132440f9d78420f82acbadfb9802ae68a8 /dev/fd0
```

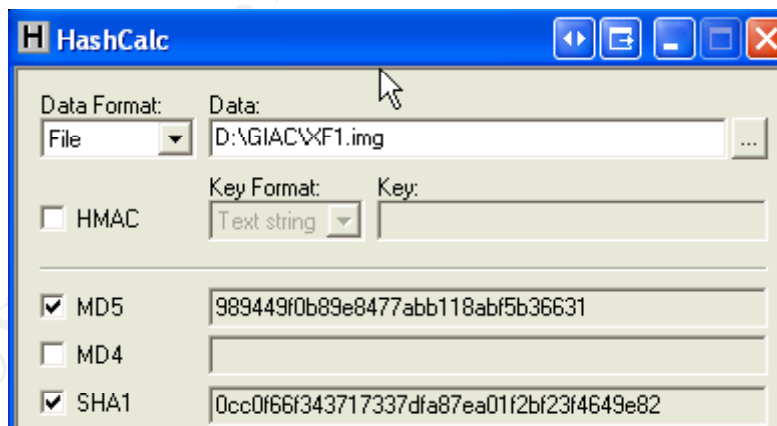
They match the baseline.

The write protection tab is set on the floppy and is moved to the Windows machine. The file is opened within X-Ways Forensics by selecting Tools -> Open Disk and then selecting the logical floppy drive option. Below is the list of available "disk" type media to choose from. We have chosen "Removable medium (A:)" as the source.



We clone this floppy disk by selecting Tools -> Disk Tools -> Clone Disk, and save the file image to XF1.img.

Once the process completes, we then run HashCalc on this new file, looking at MD5 and SHA-1 hashes.



Note that this differs from the baseline hashes. Something has changed.

The XF1.img file is copied to the USB keychain drive and transferred over to the Linux machine. First note the slight file size difference between the two files.

```
total 2916
drwxr-xr-x  2 root root    22016 Dec 31  1969 .
drwxr-xr-x  8 root root    4096 Feb  7 15:17 ..
-rwxr-xr-x  1 root root 1474560 Nov  8 16:54 fl-260404-RJL1.img
-rwxr-xr-x  1 root root 1470464 Feb  7  2005 xf1.img

[root@LinuxForensics usbdrive]#
```

We then run hexdump against each file and store the resultant output in separate text files.

```
[root@LinuxForensics usbdrive]# hexdump -v -C fl-260404-RJL1.img > /images/flhexdump.txt
[root@LinuxForensics usbdrive]# hexdump -v -C xf1.img > /images/xf1.txt
[root@LinuxForensics usbdrive]#
```

A “diff” is run between the two text files.

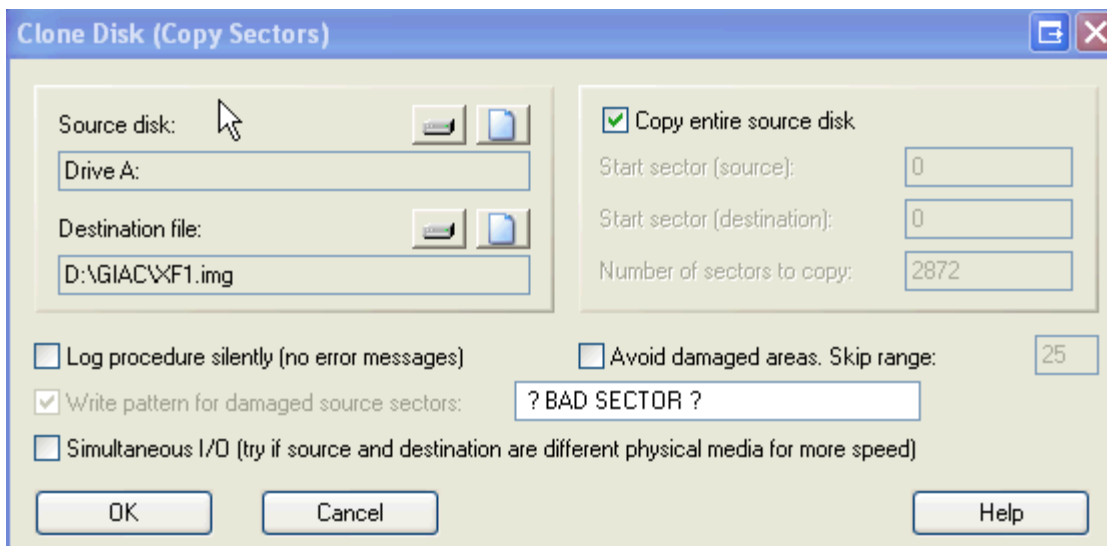
```
[root@LinuxForensics usbdrive]# diff /images/xf1.txt /images/flhexdump.txt
91905c91905,92161
< 00167000
---
> 00167000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
> 00167010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
> 00167020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
```

The 0x00 (Null) continues on...I’ve snipped the output for brevity’s sake. The last section is here...

```
> 00167fe0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
> 00167ff0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
> 00168000
[root@LinuxForensics usbdrive]#
```

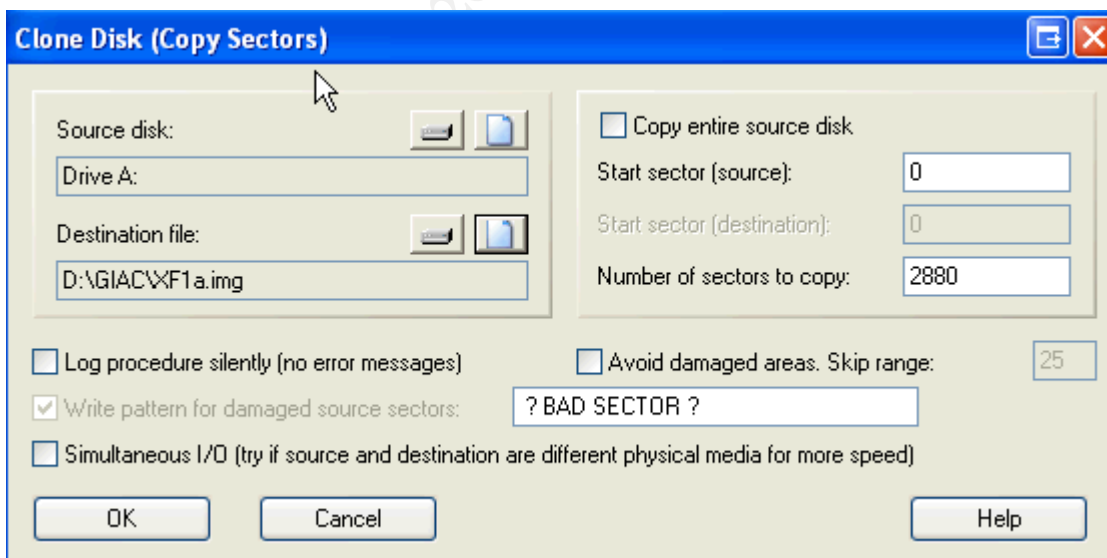
This tells us that within the fl-260404-RJL1.img file, there is an added trailing section of nulls at the end of the file which xf1.img doesn’t have. Now we must discover why X-Ways failed to copy those nulls to the image file it generated.

Looking at the options within X-Ways Forensics for disk cloning, we some items of note...

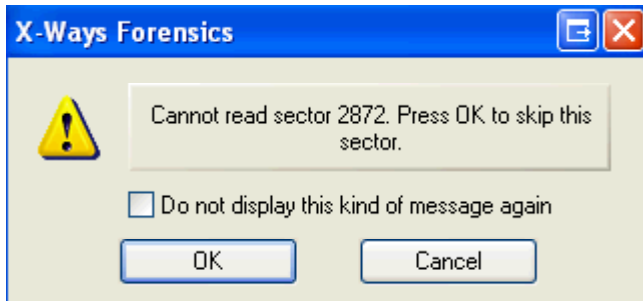


First, note that it is told to “Copy entire source disk”. The “Number of sectors to copy” was blank until I specified Drive A: as the source. Once I specified it, it read the disk, and filled in 2872. Note that back in Step 1, when we moved the image file to the physical floppy disk using dd, it indicated it had written 2880 records. It appears that X-Ways Forensics detected the unused portion of the floppy image and considered it unnecessary, truncating it on its own.

We now need to see if we can force it not to truncate the nulls found in the original. The “Copy entire source disk” is unselected, and this allows us to edit the “Number of sectors to copy”. 2880 is specified here.



Error messages are generated...

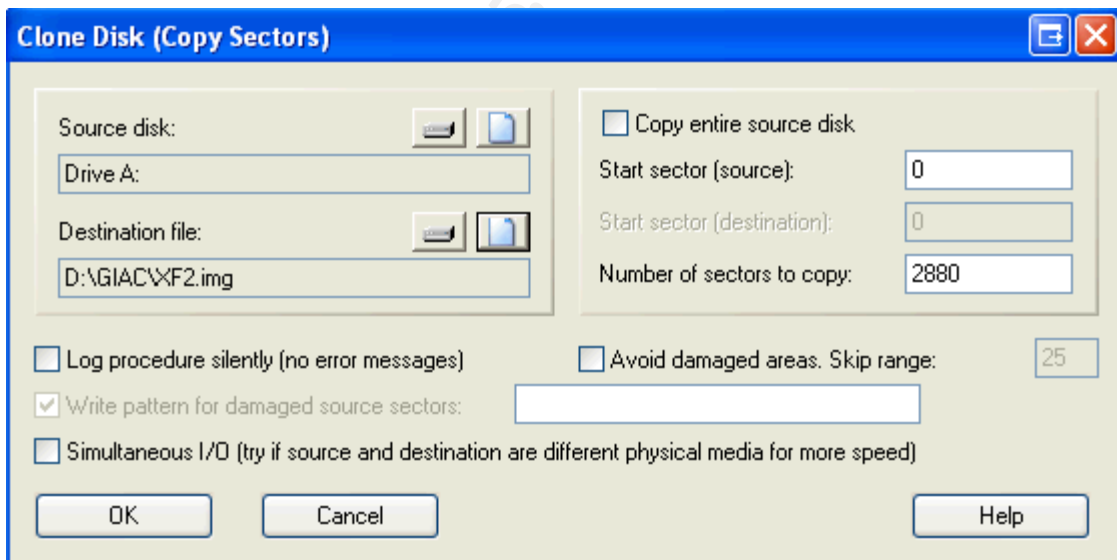


“Ok” is selected eight different times, for sectors 2872 thru 2880.

We save the resultant image to XF1a.img and open it within the X-Ways Forensics hex editor.

00166FE0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00166FF0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00167000	20 3F 20 42 41 44 20 53	45 43 54 4F 52 20 3F 20	? BAD SECTOR ?
00167010	20 3F 20 42 41 44 20 53	45 43 54 4F 52 20 3F 20	? BAD SECTOR ?
00167020	20 3F 20 42 41 44 20 53	45 43 54 4F 52 20 3F 20	? BAD SECTOR ?
00167030	20 3F 20 42 41 44 20 53	45 43 54 4F 52 20 3F 20	? BAD SECTOR ?

Note that it inserted the pattern specified above into the sectors where it didn't know what to write. Obviously, this image file will fail a pattern test also. We try again, this time not specifying anything under the write pattern.



We save the resultant image to XF2.img. Note that, this time, the image appears to be identical to the original source.

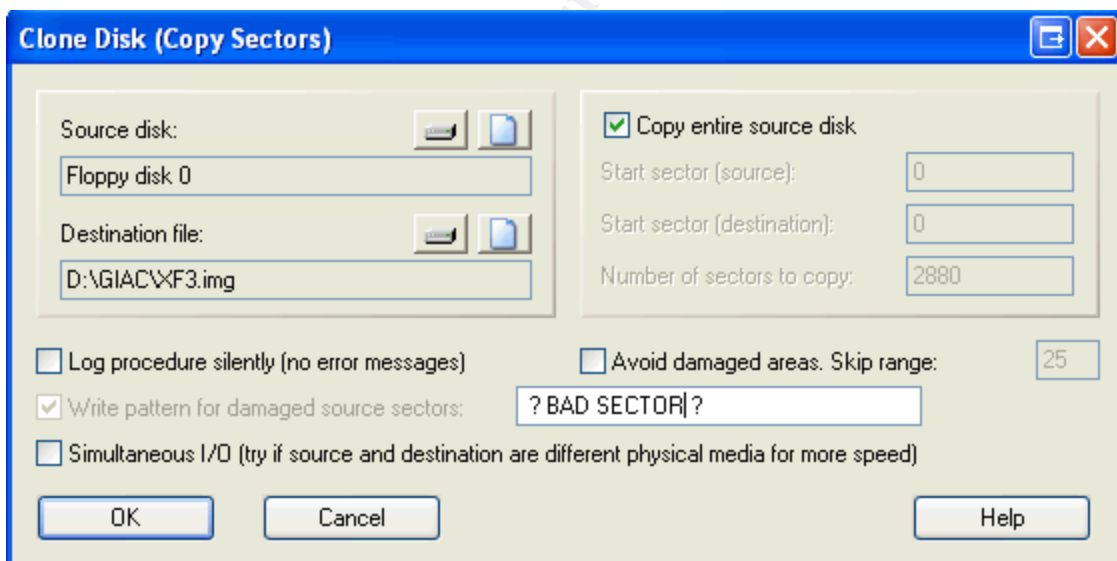
00166FE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00166FF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00167000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00167010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00167020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00167030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The hexdump/diff/md5sum/sha1sum is rerun to verify that XF2.img is identical to the original source image.

```
[root@LinuxForensics usbdrive]# diff /images/xf2hexdump.txt /images/flhexdump.txt
[root@LinuxForensics usbdrive]# md5sum ./xf2.img
d7641eb4da871d980adbe4d371eda2ad  ./xf2.img
[root@LinuxForensics usbdrive]# sha1sum ./xf2.img
20cf77132440f9d78420f82acbadfb9802ae68a8  ./xf2.img
[root@LinuxForensics usbdrive]#
```

Now it is identical.

Note that up until now, we had been using “Removable Medium (A:)”. This is shown to be a logical rather than physical selection. If we choose to rerun our test and instead of imaging from a logical disk to an image file, choose to image from the physical medium to the image file, the results are as follows...



Note here also that the program has dynamically read the disk and inserted the number “2880” as number of sectors to copy when we checked “Copy entire source disk”. Thus, it appears to be reading the same number of records as dd detected earlier.

As the program executes, we see the same errors as before about not being able to read sectors 2872 thru 2880. By selecting “Ok”, the process finishes.

A quick examination of the resultant XF3.img file shows that it has written “ ? BAD SECTOR ? ” to all bytes after 0x166FF0. Rerunning the same image, this time removing that verbiage from “Write Pattern for damaged source sectors”, results in a file that is identical to the fl-260404-RJL1.img file when MD5 and SHA-1 hashes are run on them.

Test 4 - Cloning from the original image file to disk media.

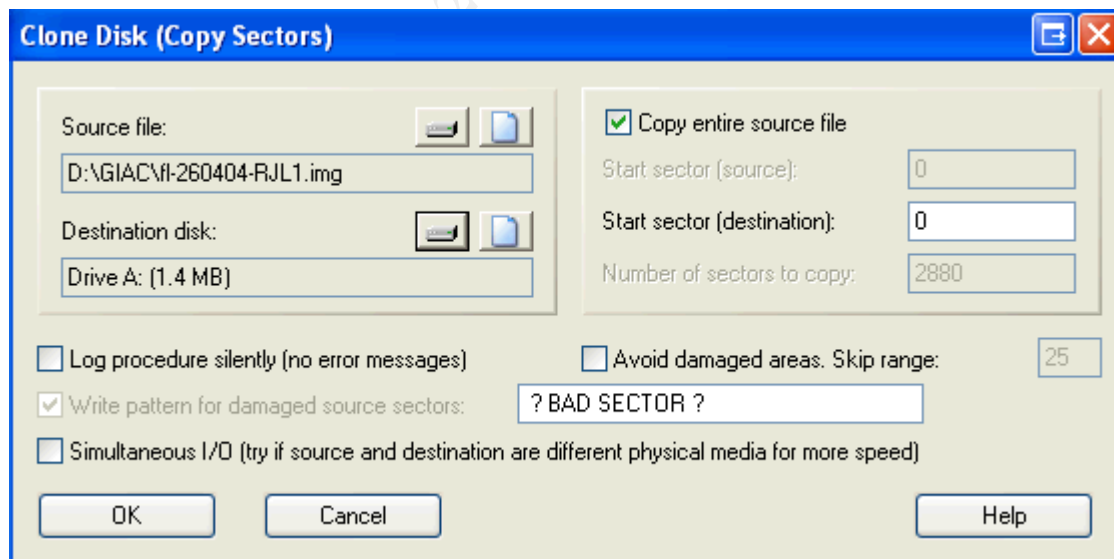
Now, we will determine how X-Ways Forensics will clone a disk from the original image file onto physical media. We confirmed that dd for Linux did this correctly in Test 3.

First, we must re-wipe the floppy disk using the same method as in Test 3.

Write protection is disabled on the floppy...

```
[root@LinuxForensics images]# dd if=/dev/zero of=/dev/fd0
dd: writing to `/dev/fd0': No space left on device
2881+0 records in
2880+0 records out
[root@LinuxForensics images]#
```

We move the floppy over to the Windows machine and perform the disk clone, selecting the logical floppy drive as our destination...



We receive the following error message



We attempt to format the floppy disk and try again...

```
d:\>format a:
Insert new disk for drive A:
and press ENTER when ready...
The type of the file system is RAW.
The new file system is FAT.
Verifying A: 44M
Initializing the File Allocation Table (FAT)...
Volume label (11 characters, ENTER for none)?
Format complete.

    1,457,664 bytes total disk space.
    1,457,664 bytes available on disk.

    512 bytes in each allocation unit.
    2,847 allocation units available on disk.

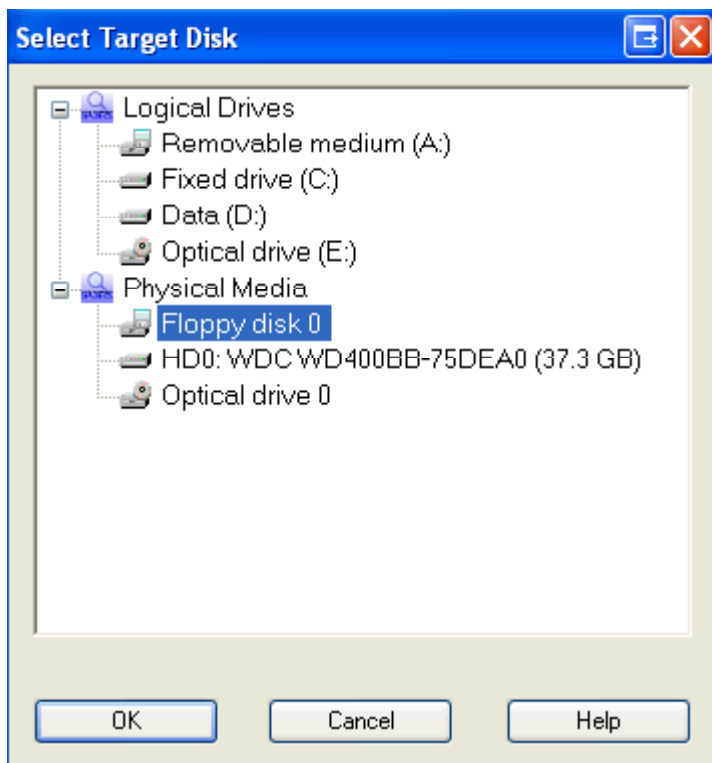
    12 bits in each FAT entry.
Volume Serial Number is 4071-BEDF
Format another (Y/N)? n
d:\>
```

This time, it successfully imaged the floppy. Enabling the write protection on the floppy and moving it to the Linux machine, we do has comparisons...

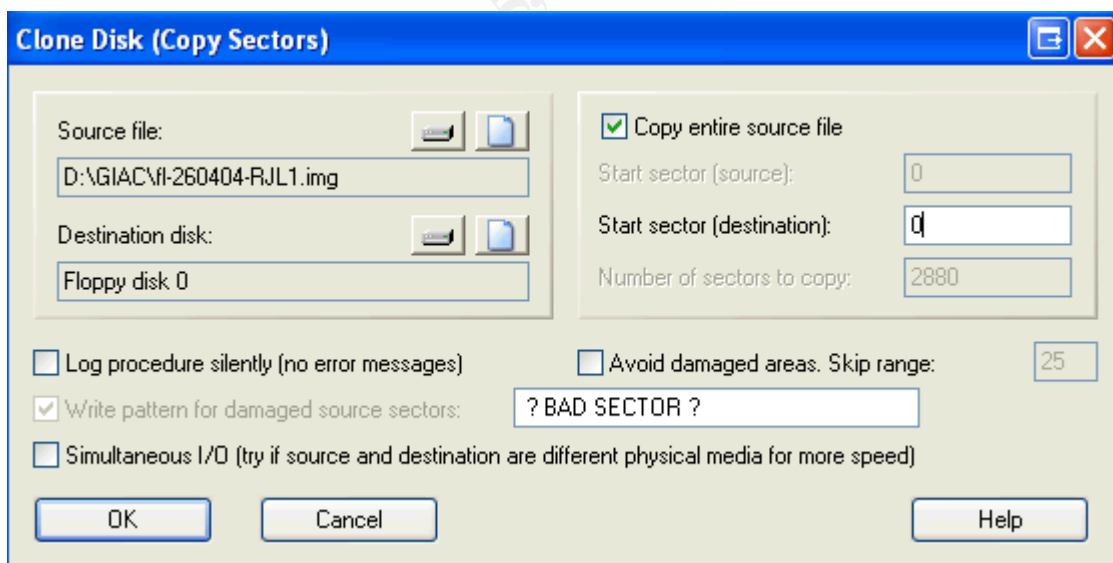
```
[root@LinuxForensics images]# md5sum /dev/fd0
d7641eb4da871d980adbe4d371eda2ad  /dev/fd0
[root@LinuxForensics images]# sha1sum /dev/fd0
20cf77132440f9d78420f82acbadfb9802ae68a8  /dev/fd0
[root@LinuxForensics images]#
```

They match to the baseline.

There are two options for transferring images from the file to the floppy disk. The first that we tried was "Removable Medium (A:)", however there is also a "Floppy Disk 0" option under "Physical Media".



If we rewrite zeroes to the floppy disk and attempt to reclone the image but this time selecting the Physical Media instead of the Logical Media, it works well.

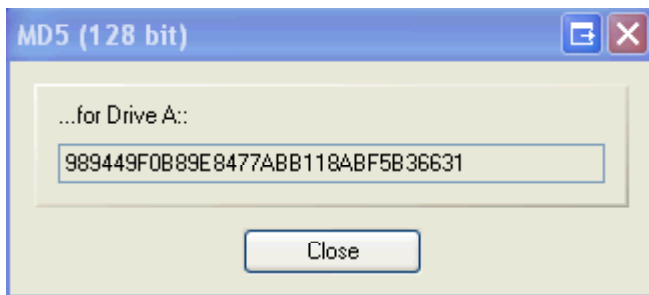


A comparison of MD5 and SHA-1 hashes show that the floppy was generated correctly.

Test 5. Hash functions of X-Ways Forensics

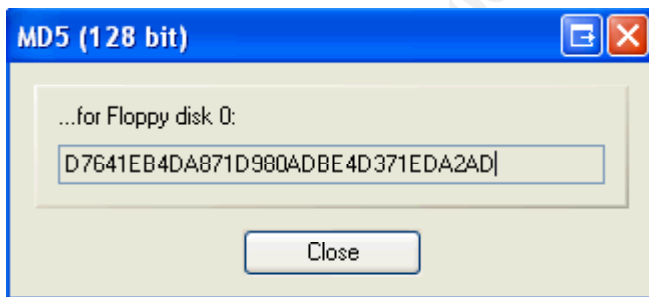
By using the created floppy disk from X-Ways Forensics, that we know to be forensically sound, we already know that md5sum on the Linux machine has correctly computed the MD5 sum.

When we open the floppy within X-Ways Forensics, and perform a Tools -> Calculate Hash, it generates a different hash than what we would expect...

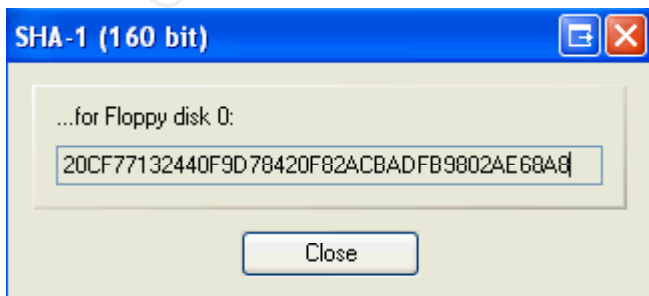


After much trial and error, it appears this is the same hash found on xf1.img (see above). Presumably, when the X-Ways Forensics program reads the floppy, it reads it as a logical drive rather than physical drive (this is an actual option when opening the media). Since it reads it as logical, it skips over those same sectors it did when creating the image file, xf1.img.

So what happens when we open the floppy disk as physical rather than logical? When we rerun the MD5 hash...



We see a match to the baseline. We run another test with SHA-1 and confirm it.



So, as long as its understood to read the media exclusively as Physical rather than Logical, X-Ways Forensics is capable of performing reliable hash functions.

Additional Analysis

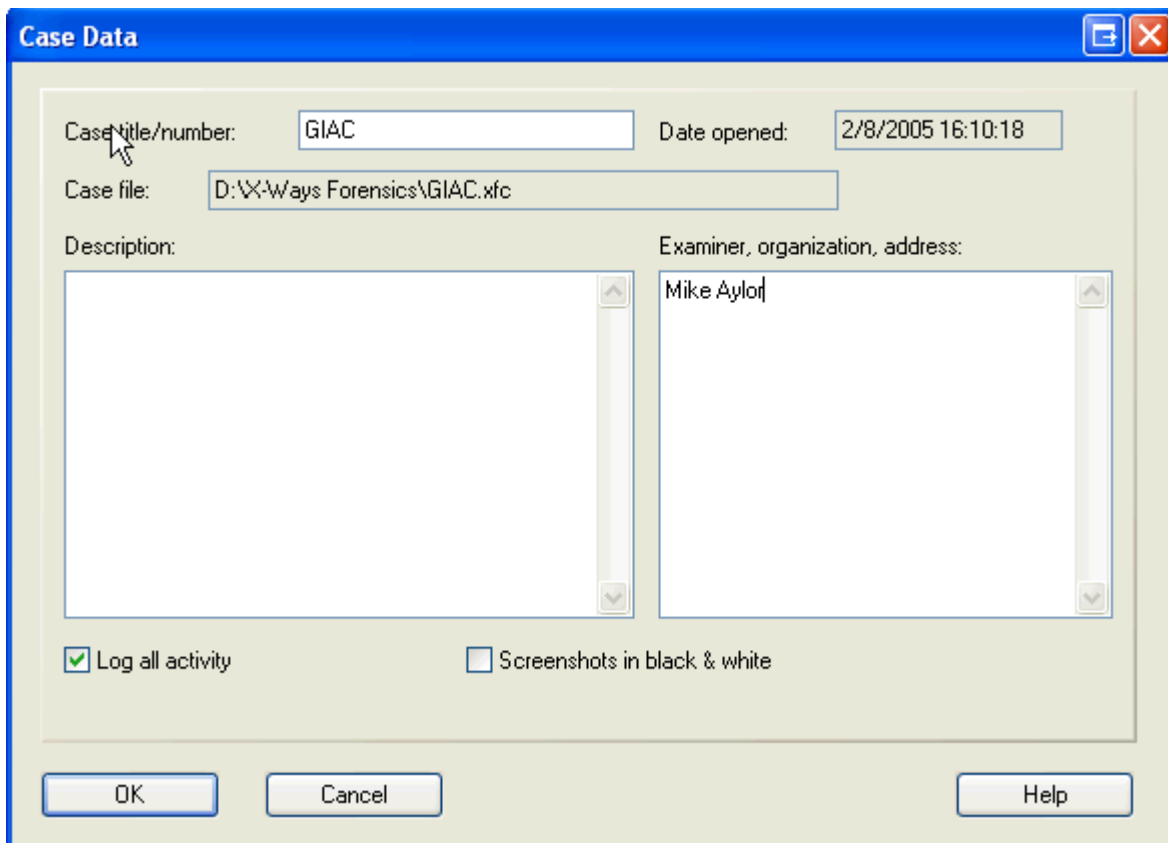
X-Ways Forensics was able to successfully clone images to physical media and physical media to images. However, it is important for the analyst who is performing the cloning to understand certain subtle effects by choosing either a logical drive or physical drive.

In order to target Logical Media, it must be able to read some kind of file table and treat it as a logical drive. This would explain why it was necessary to have to format the disk before being able to use it as a logical device within the tool. Physical Media, however, would have no such limitation, since it is simply a raw read without any attempt to “interpret” metadata, or attempt to extract information from any file allocation tables.

Presentation

One feature of X-Ways Forensics that wasn't discussed in this practical was its ability to manage cases. By opening a case file and turning on the logging, X-Ways Forensics will take snapshots of critical X-Ways Forensics screens as the analyst conducts the investigation. These are stored in the case file for future reference, and time stamped at the time the screenshot was taken.

The most critical piece of cloning disks with X-Ways Forensics is whether the forensic analyst chooses physical or logical media, and whether they want to write any tags to supposed “bad sectors”. By opening a case within X-Ways Forensics (under “Case Data”, select File -> New Case), and ensuring that all activity is logged...



Case Data

Case title/number: Date opened:

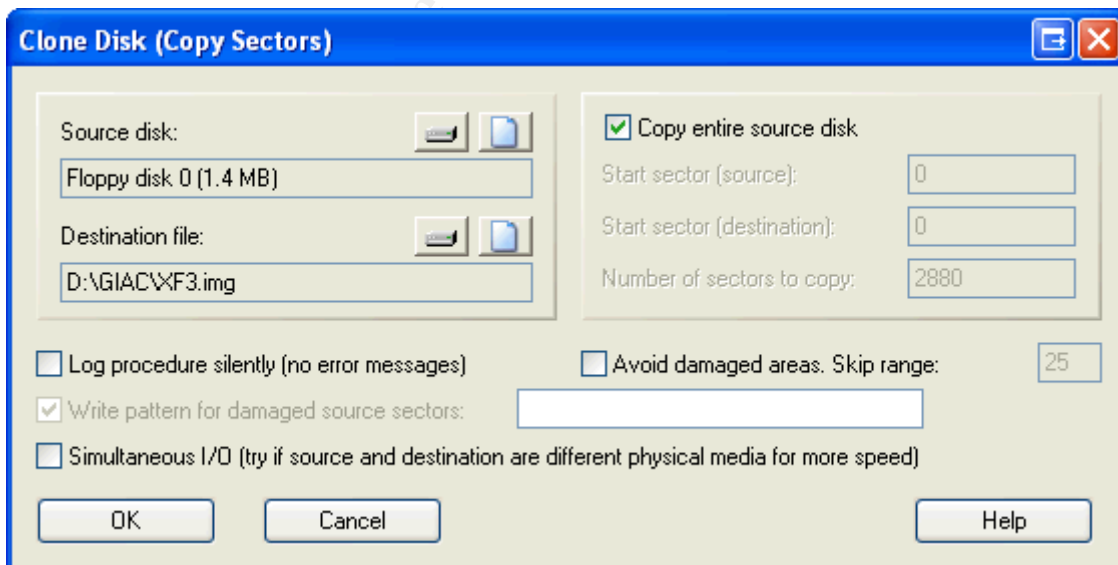
Case file:

Description:

Examiner, organization, address:

☒ Log all activity ☐ Screenshots in black & white

Subsequent screenshots will be saved to the case file and can be referenced later. For example, I reran cloning the floppy disk to an image file, and it successfully logged it by recording this screen shot...



Clone Disk (Copy Sectors)

Source disk:

Destination file:

☒ Copy entire source disk

Start sector (source):

Start sector (destination):

Number of sectors to copy:

☐ Log procedure silently (no error messages) ☐ Avoid damaged areas. Skip range:

☒ Write pattern for damaged source sectors:

☐ Simultaneous I/O (try if source and destination are different physical media for more speed)

The resultant output of image files of physical media is capable of being validated by either the built in hash calculator, or some third party tool.

Conclusion

X-Ways Forensics requires that the analyst performing the cloning understand the differences between logical and physical media, as well as understand the implications of writing a pattern to suspected bad sectors. A knowledgeable analyst would have no trouble using X-Ways Forensics to clone media in a forensically sound fashion. However, inexperienced analysts who might be used to forensic tools that require less knowledge from the user might have trouble.

Nevertheless, the testing was successful. Not only did it accurately generate image files and media that were forensically identical to one another, it's built in hash calculator was able to verify that as well.

There are many features and functions not referenced in this practical that would be invaluable to a forensic analyst, whether they were in the process of gathering the evidence from the live system or doing post-gathering analysis in a more comfortable setting.

I would recommend using the activity logging feature for case management built into the product as a good trail of what activities were performed.

© SANS Institute 2005. Author retains full rights.

List of References

8 Oct 2004, URL: <http://www.mega->

[tokyo.com/osfaq2/index.php/FAT12%20document](http://www.mega-tokyo.com/osfaq2/index.php/FAT12%20document) (11 Feb 2005).

Carrier, Brian, "Autopsy." 2005 URL:

<http://www.sleuthkit.org/autopsy/man/autopsy.html> (11 Feb 2005).

Carrier, Brian, "File System Analysis Techniques." 2005. URL:

http://www.sleuthkit.org/sleuthkit/docs/ref_fs.html (11 Feb 2005).

Fox, Jonathan, "FOXy2K: FAT System Guide." 2002. URL:

<http://home.freeuk.net/foxy2k/disk/disk6.htm> (11 Feb 2005).

Guillermi, "Breaking a steganography software: Camouflage." 16 Sep 2002

<http://http://guillermi2.net/stegano/camouflage/index.html> (11 Feb 2005).

Kessler, Gary C., "File Signatures." 9 Sep 2004.

http://www.garykessler.net/library/file_sigs.html (11 Feb 2005).

Kibria, Raihan, "frhed Homepage." URL: <http://www.kibria.de/frhed.html> (11 Feb 2005).

Slavasoft, "HashCalc – Message Digest, Checksum, and HMAC generation..."
14 Sept 2004. URL: <http://www.slavasoft.com/hashcalc/overview.htm> (11 Feb 2005).

Twisted Pear Productions. URL: <http://camouflage.unfiction.com> (11 Feb 2005).

X-Ways Software Technology AG, "WinHex/X-Ways Forensics: Installation Details." 2 Jul 2004. URL: <http://www.x-ways.net/winhex/setup.html> (11 Feb 2005)

© SANS Institute 2005, Author retains full rights.