Global Information Assurance Certification Paper

Table of Contents 1
Steven_Becker_GCFA.doc 2

Analyze an Unknown Image
and
Forensic Tool Validation: Sterilize

GCFA Practical Assignment
Version 1.5 Option 2
December 20, 2004
by
Steven Becker

Table of Contents

Table of Contents 2
Abstract: 3
Part 1 – Analyze an Unknown Image: 4
- Case Background 4
- Analysis Steps and Details 4
  - Floppy Disk Image Receipt and Verification 4
  - Enumeration of Files 7
  - Recovery of Deleted Files 8
  - File Analysis 10
  - Chasing a Possible Lead 13
  - Analyzing the Mystery Program 13
  - Finding Hidden Files 20
  - Generating a Timeline 23
  - Analyzing the Hidden Files 24
  - What Does the Evidence Show? 27
- References for Part 1 28

Part 2 – Perform Forensic Tool Validation 29
- Scope 29
- Tool Description 29
- Test Apparatus and Environmental Conditions 30
- Description of the Procedures 30
- Criteria for Approval 31
- Data and Results 32
- Analysis 37
- Presentation 37
- Conclusion 38
- References for Part 2 39

Abstract:

This practical assignment contains two main parts. The first is a mock scenario where I have been asked to analyze a floppy disk that is suspected of containing proprietary information being leaked to a competitor. This part of the assignment will explain the steps I took to analyze the data as well as the evidence recovered.

The second part of this assignment will be an analysis of a program used to create forensically sterile media. The program is called Sterilize and is available free of charge from the CyberSecurity® Institute[1].

[1] http://www.cybersecurityinstitute.biz/software/

Part 1 – Analyze an Unknown Image:

Case Background[2]

Ballard Industries is a company that designs and produces specialized fuel cell batteries used by customers around the world. Recently it seems that many customers are no longer ordering from Ballard but instead have begun to order from a competitor Rift, Inc. It seems that Rift, Inc. is now offering the same fuel cell batteries that were once unique to Ballard, Inc. A full investigation has begun to see if Rift, Inc. has somehow been given access to propriety information regarding the Ballard, Inc. fuel cell battery design and the customers who are purchasing them.

I have been asked by David Keen, the security administrator for Ballard Industries to aid in the investigation by analyzing a floppy disk seized from a Ballard Industries employee by the name of Robert John Leszczynski, Jr. This document contains a step by step explanation of how I analyzed the data and the results which I reported to Mr. Keen including the evidence discovered during my analysis.

Analysis Steps and Details

Floppy Disk Image Receipt and Verification

My analysis began by receiving a chain of custody form from Mr. Keen which contained the following information related to the seized floppy:

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

I did not receive the actually floppy disk but instead was given a compressed image of that floppy. (For the sake of this assignment an image file was downloaded here http://www.giac.org/gcfa/v1_5.gz and renamed to fl-260404-RJL.img.gz. This downloaded image is then used as if it were presented directly to me by the fictitious David Keen.)

An image file is an exact copy of all the data contained on the floppy disk, including data not accessible through normal viewing. This allows me to

[2] As described by the GIAC site http://www.giac.org/GCFA_assign_15.php

examine data that may have been deleted or hidden by Mr. Leszczynski. Compression was used to make the file smaller while is but being transported to me. Compression is easily undone and does not permanently alter the data. This is verified by comparing the digital fingerprint of the original floppy disk and the digital fingerprint of the copy I received.

The strange looking string of numbers and letters preceded by 'MD5:' on the chain of custody form is the digital fingerprint of the floppy disk seized from Mr. Leszczynski. 'MD5' is the name of the cryptographic hashing algorithm used to produce the fingerprint and followed by the name of the data the fingerprint represents, fl-260404-RJL1.img.

Digital fingerprints are referred to as 'hashes' of the data. To create a digital fingerprint the data is put through a mathematical algorithm that creates a unique numerical representation of the original data. In this case the algorithm used was the Message Digest 5 (MD5). Hash values are especially useful for detecting changes in data because the algorithms are created so that even a small change in data will create a noticeable change in the hash value.

According to RFC1321:

> "...the difficulty of coming up with two messages having the same message digest is on the order of 2^64 operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^128 operations."[3]

This means that the likelihood of two sets of data resulting in the same digital fingerprint is 1 in 2^64 (1 in 18,446,744,073,709,551,616).

I took the compressed image file, which is an exact copy of the seized floppy, named fl-260404-RJL1.img.gz and transferred it to my forensic analysis machine. As stated earlier the compression simply makes the image file smaller so that it takes up less space when transferring it and is completely reversible. After transferring the image to my workstation I uncompressed it, returning it to its original state and took a digital fingerprint to verify that it had not been modified. Below is a screen capture from my forensic workstation showing that the resulting MD5 hash (fingerprint) matches the value provided to me by Mr. Keen.

[3] http://www.faqs.org/rfcs/rfc1321.html

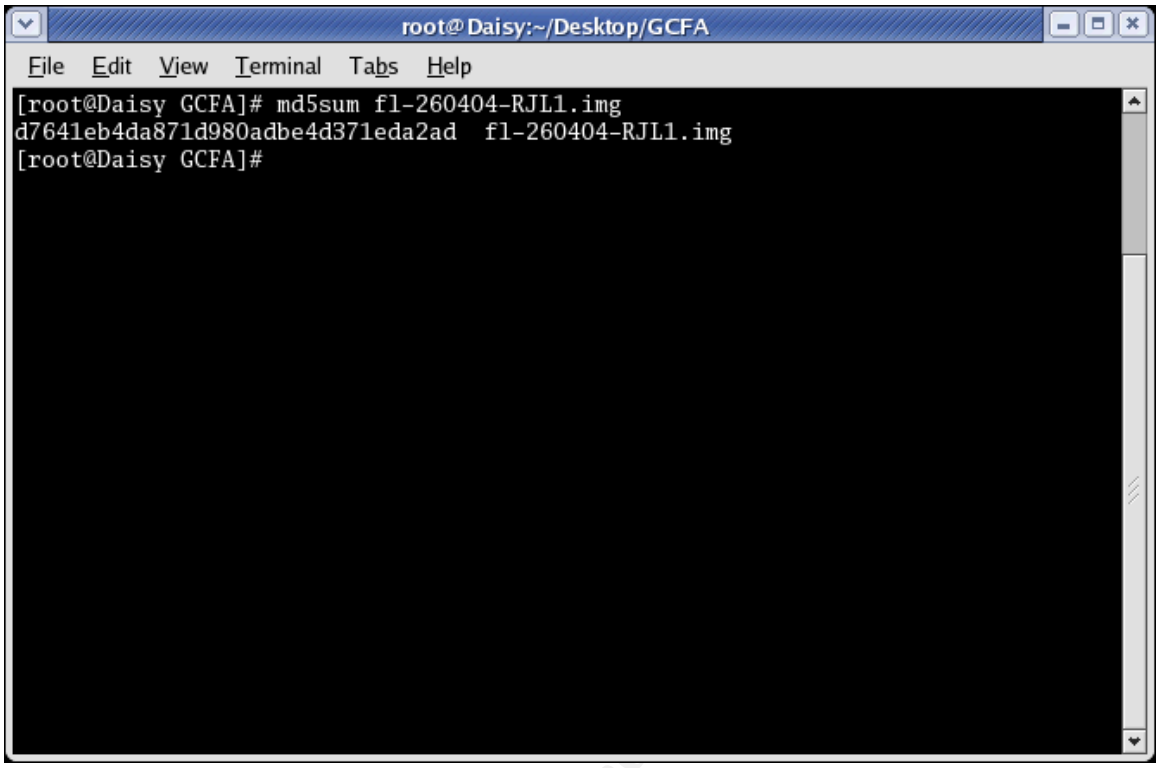


Figure 1

Note that the hash value of the image file shown in figure 1 matches the hash value of the seized floppy given to us by Mr. Keen listed on the chain of custody form as:

d7641eb4da871d980adbe4d371eda2ad

After verifying that my image of the floppy disk was forensically identical to the original floppy disk, I mounted it using the command:

mount -o loop,ro /root/Desktop/GCFA/fl-260404-RJL1.img /mnt/evidence_mount/

This command basically allows me to access the data in the image the same way I could if I had the original floppy disk in the floppy drive of my workstation. The only difference is that I mounted it using the 'ro' option which stands for 'read-only'. By mounting it this way I am prevented from accidentally modifying the data and tainting the evidence while analyzing it. Even though I will actually be examining the image I will refer to it as 'the floppy' since that's where the data was originally.

Enumeration of Files

The next several steps used utilities distributed with The Sleuth Kit (TSK)[4] which is a collection of tools used for computer investigations. The first step was to use the tool called 'fls' to list the files on the floppy and details about them. The exact command I used was:

```
fls -l -f fat12 -z MDT7MST /root/Desktop/GCFA/fl-260404-RJL1.img
```

The '-l' flag tells the utility that we want the long version of output which includes information about where the entry physically sits on the disk, the name of the entry, the modified, accesses, changed (MAC) times related to the files, the size of the files, and who the owns the files. The '-f' option tells the utility what type of file system it's examining (fat12 is the floppy disk file system). The final flag '-z' lets us tell the utility the time zone in which the activity took place, in this case Mountain Time.

The results contain 9 pieces of data for each entry.

1. File type - The file type tells us if it's a file or a directory, normal files are represented by 'r/r' and directories would be shown as 'd/d'.

2. Inode - The inode is simply an address on the disk where the metadata resides for that file. Metadata is data about data. In the case of computer disks the metadata tells us about the data and files on the disk, such as where on the disk the pieces of a file reside, the size of the file, the last time the file was used, etc. Any inode with a '*' represents a deleted file.

3. File Name - Is the name of the file in two formats. The first is the long format and the second (contained in parentheses) is a shortened form for backward compatibility with older systems.

4. Modification Time - This is the time that the file was last written to or modified.

5. Access Time - This is the time that the file was last read or accessed.

6. Change/Created Time - This is the time that the inode information (metadata) for the file was last modified.

7. Size - The number of disk blocks containing the file.

8. User ID* - The system ID of the user who owns the file.

---

* Floppy disks do not support users and groups so the User IDs and Group IDs are set to zero.

[4] The Sleuth Kit is available at http://www.sleuthkit.org/sleuthkit/index.php

9. Group ID* - The system ID of the group to which the file belongs.

The results were saved to a spreadsheet shown below in figure 2.

file type	inode	file name	modified time	accessed time	change time	size	user id	group id
r/r	3:	RJL (Volume Label Entry)	2004.04.25 10:53:40 (MST)	2004.04.25 00:00:00 (MST)	2004.04.25 10:53:40 (MST)	0	0	0
r/r	* 5:	CamShell.dll (_AMSHELL.DLL)	2001.02.03 19:44:16 (MDT)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:18 (MST)	36864	0	0
r/r	9:	Information_Sensitivity_Policy.doc (INFORM~1.DOC)	2004.04.23 14:11:10 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:20 (MST)	42496	0	0
r/r	13:	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)	2004.04.22 16:31:06 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:22 (MST)	32256	0	0
r/r	17:	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)	2004.04.22 16:31:06 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:24 (MST)	33423	0	0
r/r	20:	Password_Policy.doc (PASSWO~1.DOC)	2004.04.23 11:55:26 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:26 (MST)	307935	0	0
r/r	23:	Remote_Access_Policy.doc (REMOTE~1.DOC)	2004.04.23 11:54:32 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:36 (MST)	215895	0	0
r/r	27:	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)	2004.04.23 14:10:50 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:46:44 (MST)	22528	0	0
r/r	* 28:	_ndex.htm	2004.04.23 10:53:56 (MST)	2004.04.26 00:00:00 (MST)	2004.04.26 09:47:36 (MST)	727	0	0

Figure 2

These results show that there are nine entries on the floppy, the first one, at inode 3, is the entry for the name of the floppy. The entries at inodes 5 and 28 are deleted files, which I made note of so that I could recover them and examine them. The remaining entries are for files that all have a .doc extension and appear to be regular and accessible through normal means.

One interesting thing that caught my eye was that two of the regular (not deleted) files are an order of magnitude larger then the other four. This means that they are either very long documents or possibly being used to conceal contraband information.

A quick internet search for information on the deleted file CamShell.dll using Google™[5] turned up a discussion which indicated that CamShell.dll may be related to a program called Camouflage[6]. I decided to do more research on this program after further analysis of this deleted file and the other.

Recovery of Deleted Files

Next I proceeded to recover the two deleted files. In order to do this I needed to determine where on the floppy disk their data resided. This was done using the tool 'istat' from The Sleuth Kit on the inode of each file:

[5] http://www.google.com
[6] http://www.tranceaddict.com/forums/archive/topic/79627-1.html

```
istat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 5

istat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 28
```

Just like when using the 'fls' command earlier we use the -f flag to tell the utility that it's examining a floppy disk. The '5' at the end of command tells the 'istat' tool that we are wanting information for the file with inode 5 which, we see from figure 2, is the deleted file CamShell.dll. Likewise the '28' at the end of the second 'istat' command tells the tool we want information related to the deleted file named '_ndex.htm'. The output from running this command on each inode is shown below:

```
[root@Daisy GCFA]# istat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 5
Directory Entry: 5
Not Allocated
File Attributes: File, Archive
Size: 36864
Num of links: 0
Name: _AMSHELL.DLL

Directory Entry Times:
Written:   Sat Feb  3 19:44:16 2001
Accessed:  Mon Apr 26 00:00:00 2004
Created:   Mon Apr 26 09:46:18 2004

Sectors:
33

Recovery:
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96
97 98 99 100 101 102 103 104
[root@Daisy GCFA]# istat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 28
Directory Entry: 28
Not Allocated
File Attributes: File, Archive
Size: 727
Num of links: 0
Name: _ndex.htm

Directory Entry Times:
Written:   Fri Apr 23 10:53:56 2004
Accessed:  Mon Apr 26 00:00:00 2004
Created:   Mon Apr 26 09:47:36 2004

Sectors:
33

Recovery:
33 34
```

Figure 3

The important details we need for recovering the deleted files are the Recovery sectors. This tells us which sectors on the disk contain the data for the file we wish to recover. In this case it appears that one file was written over the top of the other. We know this because the Recovery sectors for _ndex.htm are the same as the first two recovery sectors for CamShell.dll. Based on the times that the files were written to disk (the Change times as listed in Figure 2) my guess would be that the _ndex.htm file was written over Camshell.dll.

The next step in recovering the files is to copy the data out of the Recovery

sectors for each one out to a new file. This was done using the 'dcat' utility from The Sleuth Kit which simply copies the contents of a particular data unit. We are able to tell the 'dcat' utility what sector to start at and how many sectors to copy. The output in Figure 3 shows us that the Recovery sectors for CamShell.dll start at 33 and are contiguous to 104 which is 72 sectors total. To recover CamShell.dll I used the following command:

```
dcat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 33 72 > CamShell.dll.recovered
```

The two numbers following the path to the floppy disk image are the starting sector for the data and the total sectors needing to be copied. The greater than symbol sends the data to the file named CamShell.dll.recovered instead of just displaying it all on the screen. This process was repeated for '_ndex.htm' with that starting sector being 33 again and the total sectors to be copied being 2:

```
dcat -f fat12 /root/Desktop/GCFA/fl-260404-RJL1.img 33 2 > _ndex.htm.recovered
```

File Analysis

Next I used the Linux command 'file'[7] to determine what kind of files these may be. This command reported that they were both 'HTML document text'. Since they were reported as being text documents I used the Linux command 'less' to read them. 'Less' simply allows the user to read the contents of a file. It became apparent that _ndex.htm was indeed a HTML document as reported by the 'file' command. It also became apparent that CamShell.dll only contained HTML text at the beginning of the file and that the rest contained unreadable data interspersed with readable words much like an executable program does. This furthered my speculation that the file _ndex.htm had overwritten the file CamShell.dll. The HTML recovered from the two files is shown below:

[7] For information on the 'file' command and other Linux commands used please visit http://linux.ctyme.com/

```
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html;  charset=ISO-8859-1">
<TITLE>Ballard</TITLE>
</HEAD>
<BODY bgcolor="#EDEDED">

<center>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
 WIDTH="800" HEIGHT="600" id="ballard" ALIGN="">
 <PARAM NAME=movie VALUE="ballard.swf"> <PARAM NAME=quality VALUE=high> <PARAM NAME=bgcolor
VALUE=#CCCCCC>
 <EMBED src="ballard.swf" quality=high bgcolor=#CCCCCC  WIDTH="800" HEIGHT="600" NAME="ballard" ALIGN=""
 TYPE="application/x-shockwave-flash" PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
</center>
</BODY>
</HTML>
```

Figure 4

This HTML appears to be a simple webpage that attempts to load a Shockwave™ Flash movie which was not found on the floppy disk. At this point in the analysis the HTML seemed uninteresting as far as potentially being used to leak proprietary information out of Ballard Industries so I refocused on finding more about the rest of the CamShell.dll file.

The next tool I used to examine CamShell.dll was the Linux command 'strings'. This command searches a file and displays the printable characters. It is very helpful in pulling the human readable words from files that are mostly machine code much like the bulk of CamShell.dll appeared to be. The output of the 'strings' command was saved to a file using the command:

```
strings CamShell.dll.recovered > CamShell.dll.strings
```

I used the Linux command 'less' again to examine the output from the strings command. There were several interesting strings of text contained in the file. A partial list is shown below:

```
advapi32.dll
CamouflageShell
CamShell
CamShell.dll
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb
C:\WINDOWS\SYSTEM\MSVBVM60.DLL\3
DeleteDC
DeleteObject
DllCanUnloadNow
DllFunctionCall
DllGetClassObject
DllRegisterServer
DllUnregisterServer
DragQueryFileA
EVENT_SINK2_AddRef
EVENT_SINK2_Release
EVENT_SINK_AddRef
EVENT_SINK_QueryInterface
EVENT_SINK_Release
FindResourceA
FIShellExtInit
GetFullPathNameA
GetObjectA
GetTextMetricsA IContextMenu
IContextMenu_GetCommandString
IContextMenu_InvokeCommand
IContextMenu_QueryContextMenu
IContextMenu_TLB
IctxMenu.tlbWW
idCmd
idCmdFirst
idCmdLast
indexMenu
InsertMenuA
IShellExtInit
IShellExtInit_Initialize
LoadBitmapA
LoadLibraryA
LoadResource
MSVBVM60.DLL
RegCloseKey
RegOpenKeyExA
RegQueryValueExA
ReleaseStgMedium
RtlMoveMemory
SelectObject
SetMenuItemBitmaps
shell32.dll
Shell_Declares
ShellExt
Shell_Functions
stdole2.tlbWWW
StretchBlt
StringFromGUID2
SystemParametersInfoA
```

Figure 5

I searched for many of the strings using Google™ and found that many of them were Microsoft® Windows system files and Microsoft® Visual Basic® files or function calls. I noticed several lines that contained the word 'Camouflage' which I had seen reference too when searching the Internet for information on this file.

Next I examined the six files on the floppy disk that had not been deleted. Since they had not been deleted there was no need to extract them using The Sleuth Kit tools. I simply navigated to the mounted read only floppy image at /mnt/evidence_mount. From there I ran 'file' against each of them to see what they reported to be. They all reported to be Microsoft Office Documents. This being the case I decided to open them with a word processor on an isolated machine. I used an isolated machine in case the documents happened to contain a virus or other malware.

They all opened with Microsoft® Word and appeared normally in the word processor. They all appeared to be policy files based on the example policies and templates available from SANS[8]. The two files that I noted as being very large earlier in the analysis were each only three pages long which led me to suspect that there could be data hidden in them.

Chasing a Possible Lead

The next step was to see if I could tie the reference to Camouflage and the large Word documents together. To do this I started by doing a Google™ search for "hiding data in word documents using camouflage". This search resulted in several interesting hits. The first page I looked at[9] was an article discussing how the author found data hidden in a picture that was put there using a program called Camouflage. While the article itself was very interesting, what was more useful to this investigation were the links at the bottom of the page which linked to tools to unprotect Camouflage files. I took note of these links in case it turned out that Camouflage was indeed used and the files were password protected. I went back to my search page and looked for sites that contained copies of the Camouflage program for download. I was fortunate to find and download a copy of version v1.2.1 of the file[10] which I installed on my sandbox machine.

Camouflage is a program that hides a file or file within another file. The website from which I downloaded my copy used for testing described it this way:

[8] http://www.sans.org/resources/policies/#template
[9] http://guillermito2.net/stegano/camouflage/
[10] Camouflage v1.2.1 was downloaded from http://camouflage.unfiction.com/

"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."[11]

Analyzing the Mystery Program

The next step was to verify that Camouflage v1.2.1 is the same program that the deleted CamShell.dll file from the floppy is a part of. For comparison I copied the CamShell.dll file from my sandbox machine to my forensics workstation. If both CamShell.dll files were from the same version of the Camouflage program, then the MD5 hashes should be the same as should a MD5 hash of any part of the files. I knew from earlier examination that the first part of the recovered CamShell.dll contained the HTML text from the recovered file _ndex.htm. I knew that for this reason an MD5 hash of the CamShell.dll files would not be the same since the recovered CamShell.dll had been altered. I decided I would make a copy of the recovered CamShell.dll removing the first part which had been overwritten by _ndex.htm. I also made a copy of the control CamShell.dll from the Camouflage installation I downloaded from the Internet. I removed from this copy the same number of bytes that had been changed from the beginning of the recovered CamShell.dll.

To do this I loaded the _ndex.htm file and each CamShell.dll into a hex editor. I found that _ndex.htm was 1024 bytes long, numbered as bytes 0-1023. I switched to the CamShell.dll.recovered file and selected byte number 1024 through the end of the file, byte number 36863, and copied those bytes to a file named CamShell.dll.recovered.partial. I did the same using the CamShell.dll.control file and saved those bytes to CamShell.dll.control.partial. I then ran md5Sum against all the files and found that the hashes for CamShell.dll.recovered.partial and CamShell.dll.control.partial were the same indicating that the recovered CamShell.dll was originally part of a Camouflage v1.2.1 installation. A screen shot of the dm5sum results is shown below:

[11] http://camouflage.unfiction.com/

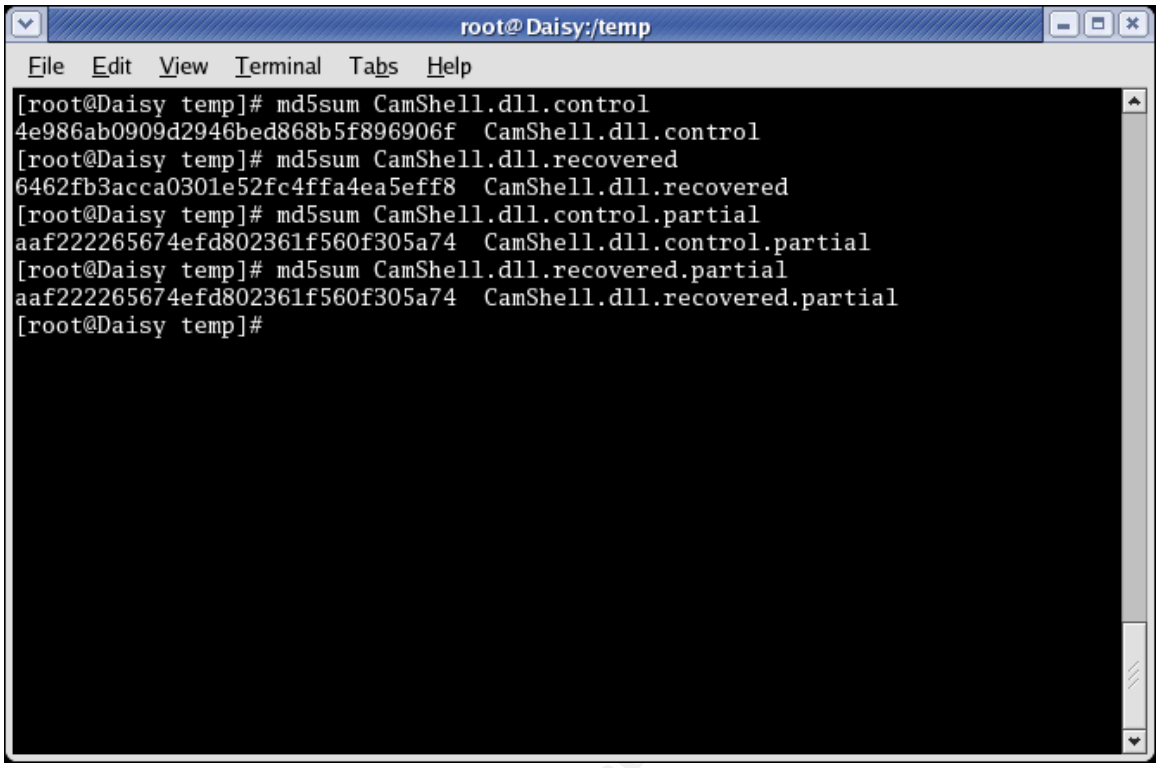


Figure 6

Camouflage v1.2.1 was downloaded as a self extracting zip file. When extracted the program installed four files into C:\Program Files\Camouflage. Those files were:

Camouflage.exe
CamShell.dll
Readme.txt
Uninst.isu

Executing the Camouflage.exe program brought up a settings dialog that allows the user to configure the behaviors of Camouflage such as how menu options are named and what file attributes are shown. Below is an example of the settings dialog:

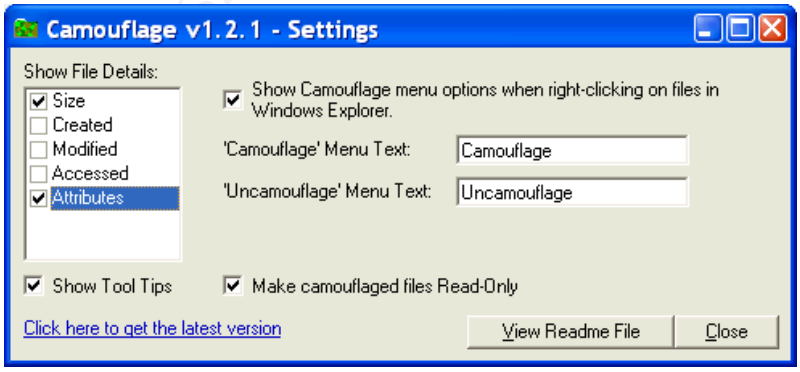


Figure 7

The CamShell.dll is the file which does the actual work. To use it simply right click a file, or files, with your mouse. The user is then presented with the usual menu options as well as two new ones, Camouflage and Uncamouflage (figure 8).

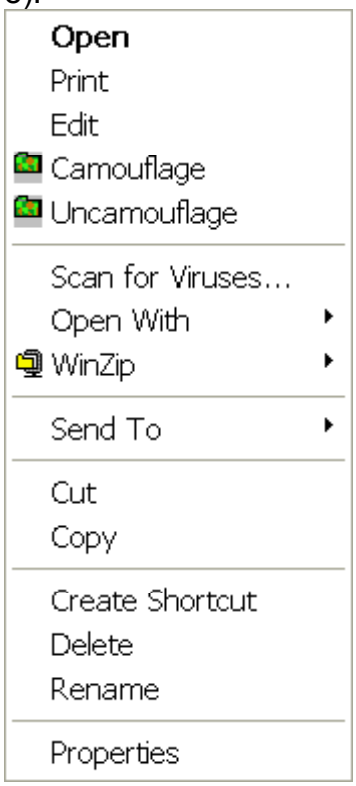


Figure 8

When the Camouflage option is selected a dialog box pops up showing the file or files that have been selected to be hidden:

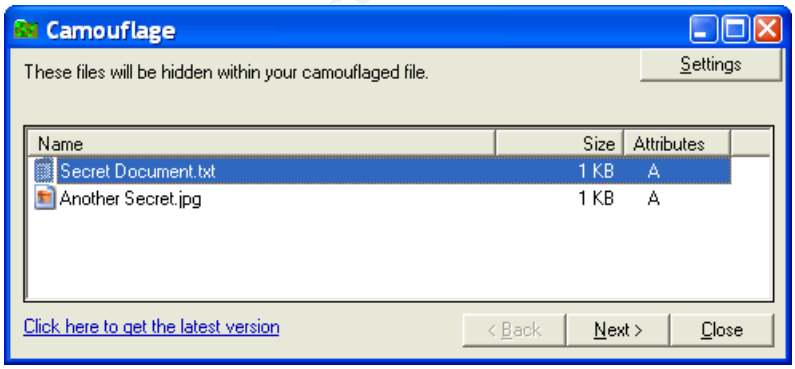


Figure 9

Next the file that will be used to host the Camouflaged data is selected:

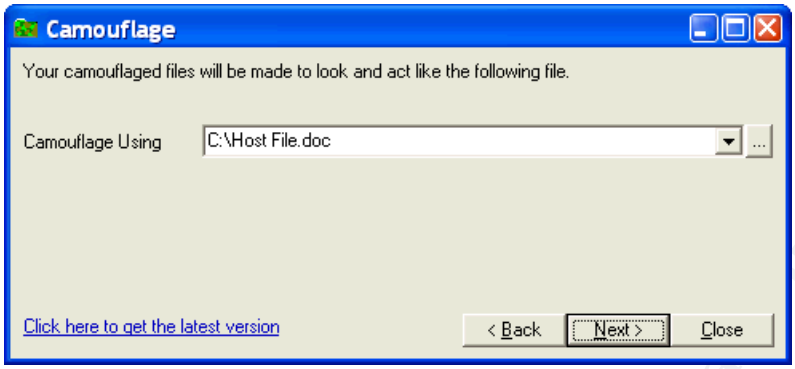


Figure 10

Next the location and filename for the new Camouflage file is entered. The read only option is given since altering a Camouflaged file will destroy the hidden data. (This was verified experimentally):

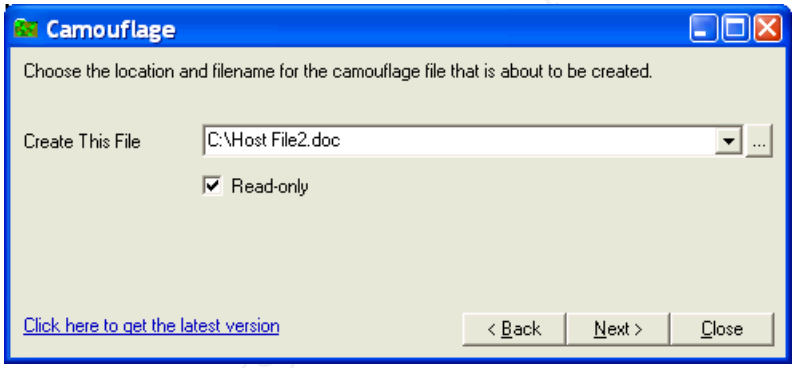


Figure 11

Finally the option to provide a password is given:

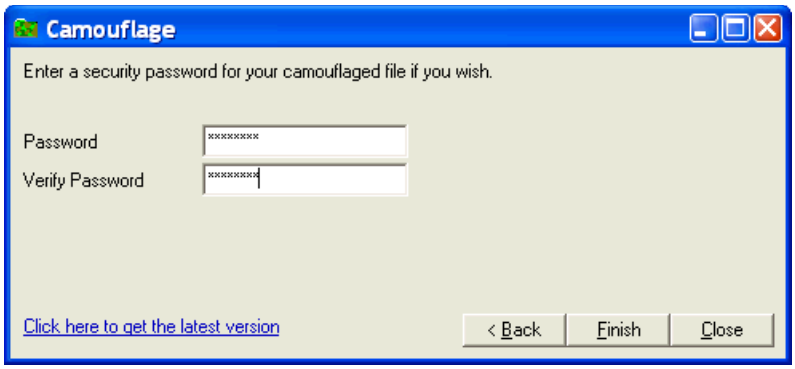


Figure 12

The resulting file, in this case C:\Host File2.doc, opens normally using Microsoft® Word. The only noticeable difference is that the new camouflage file is larger then the original host document.

To Uncamouflage a file the procedure is basically the reverse of the Camouflaging it. Simply right click the Camouflaged file and select 'Uncamouflage' from the menu (Figure 8). At that time the password dialog box appears and prompts the user to enter the password:

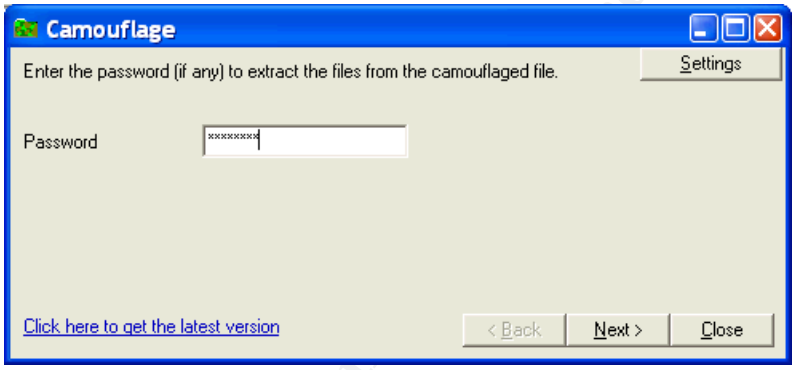


Figure 13

If the password is correct then a dialog box appears showing what files are hidden and what the original host file was called. The hidden files are indicated by the Camouflage icon:

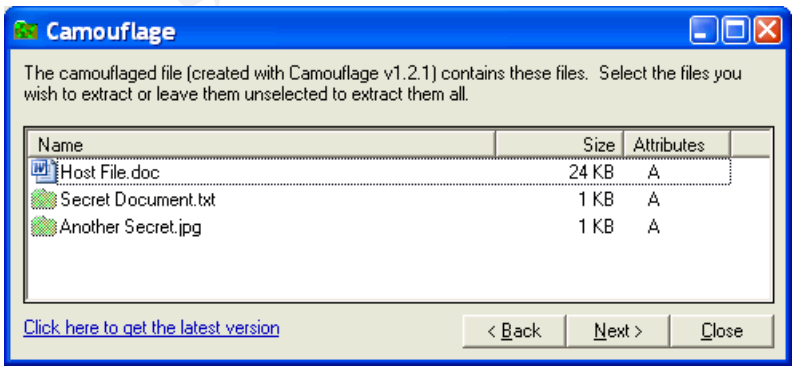


Figure 14

After selecting which files to extract a dialog is presented for the user to select where the files should be recovered to:

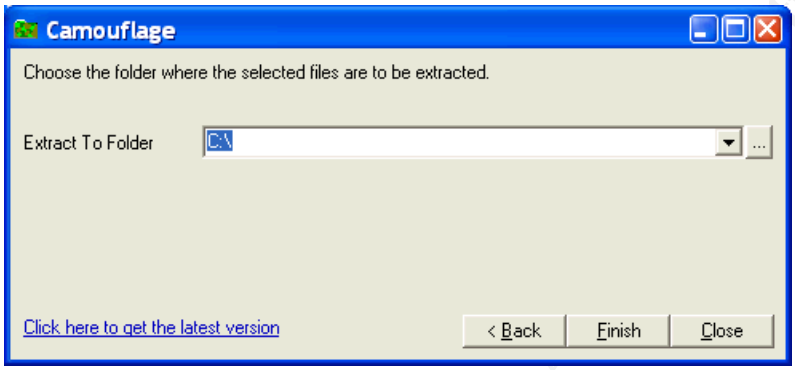


Figure 15

The Uncamouflaged files now appear the same as before they were hidden. A quick 'md5sum' of the original files and the Uncamouflaged files show that using Camouflage does not change the files at all since they return the same MD5 hash value.

The Camouflage 'Readme.txt' files is, as the name suggests, a read me file containing instructions on how to use Camouflage and the changes made in this release (v1.2.1). Lastly the 'Uninst.isu' file is used when uninstalling the program.

There were also several registry entries made during installation of Camouflage v1.2.1. Some of the more interesting ones included:

```
HKEY_CLASSES_ROOT\*\shellex\ContextMenuHandlers\Camouflage
HKEY_CLASSES_ROOT\CamouflageShell.ShellExt
HKEY_CLASSES_ROOT\CLSID\{29557489-990B-11D4-9413-004095490AD4}
HKEY_CURRENT_USER\Software\Camouflage
HKEY_CURRENT_USER\Software\Camouflage\CamouflageFile
HKEY_CURRENT_USER\Software\Camouflage\OutputFile
HKEY_CURRENT_USER\Software\Camouflage\OutputFolder
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MenuOrder\Start
    Menu2\Programs\Camouflage
HKEY_CURRENT_USER\Software\Microsoft\Windows\ShellNoRoam\MUICache
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\*\shellex\ContextMenuHandlers\Camouflage
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CamouflageShell.ShellExt
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{29557489-990B-11D4-9413-
    004095490AD4}
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App
    Management\ARPCache\Camouflage
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App
    Paths\Camouflage.exe
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\Camouflage
HKEY_LOCAL_MACHINE\SOFTWARE\Twisted Pear Productions\Camouflage\1.2.1
HKEY_USERS\S-1-5-21-1078081533-1614895754-1801674531-
    1338\Software\Camouflage\CamouflageFile
HKEY_USERS\S-1-5-21-1078081533-1614895754-1801674531-
    1338\Software\Camouflage\OutputFile
HKEY_USERS\S-1-5-21-1078081533-1614895754-1801674531-
    1338\Software\Camouflage\OutputFolder
HKEY_USERS\S-1-5-21-1078081533-1614895754-1801674531-
    1338\Software\Microsoft\Windows\CurrentVersion\Explorer\MenuOrder\Start
    Menu2\Programs\Camouflage
```

Figure 16

Since general users don't know about the registry or don't know how to modify it monitoring the registry for these entries could be very useful. A tool that monitors the registry for changes such as Tripwire® could be used to watch for the addition of one or more of these registry keys. There are several keys that are especially interesting for investigative purposes. Specifically

HKEY_CURRENT_USER\Software\Camouflage\CamouflageFile
HKEY_CURRENT_USER\Software\Camouflage\OutputFile and
HKEY_CURRENT_USER\Software\Camouflage\OutputFolder

The values in the CamouflageFile key indicate what files have been used to host Camouflaged data. The values in the OutputFile key list the resulting Camouflage files and where they were saved. The OutputFolder lists the directories where Camouflaged data has been uncamouflaged to. It would be interested to view the registry of Mr. Leszczynski's machine to see if the file I have recovered also appears in his Camouflage related registry entries. This would help show that it was in fact Mr. Leszczynski that hid these files and therefore help show that he was knowingly removing them from the company grounds in violation of Ballard Industries policy.

I searched the registry on my sandbox machine to see if any entries existed that listed what files were actually hidden using Camouflage. I was unable to find any entries that listed this information.

Finding Hidden Files

I systematically attempted to open each of the six Word documents using Camouflage. Camouflage prompted me for a password for each file. Not having recovered any passwords yet I tried leaving it blank and was able to open one file with Camouflage. The file 'Internal_Lab_Security_Policy.doc' contained a hidden text file named 'Opportunity.txt' which contained the following text:

> I am willing to provide you with more information for a price. I have included a sample of our Client Authorized Table database. I have also provided you with our latest schematics not yet available. They are available as we discussed - "First Name".
> My price is 5 million.
>
> Robert J. Leszczynski

Figure 17

The remaining five files were either password protected or did not have data hidden in them with this version of Camouflage. This being the case I decided to look into finding a utility to help open password protected Camouflage files.

I returned to the web site I had found previously and looked again at the links at the bottom of the page. The one that seemed most promising was a program called CamoDetect written by Andrew Christensen[12]. CamoDetect is a Perl script that claims to be able to detect if Camouflage has been used to hide data in a particular file and if a password had been used it can retrieve it. I downloaded this file and ran it against each of the six Word documents. I ran it against the document we had already retrieved hidden data from to verify that the program actually worked as described.

The CamoDetect script, called 'SetecAstronomy.pl' was able to determine that three files were if fact hiding data using Camouflage. It was able to determine the amount of hidden data in bytes, the number of hidden files in each document, and that two of them were password protected. The script was able to determine the password for the two protected files even made an unprotected copy of each. The output from running the script is shown below:

[12] http://packetstormsecurity.nl/crypt/stego/camouflage/SetecAstronomy.pl

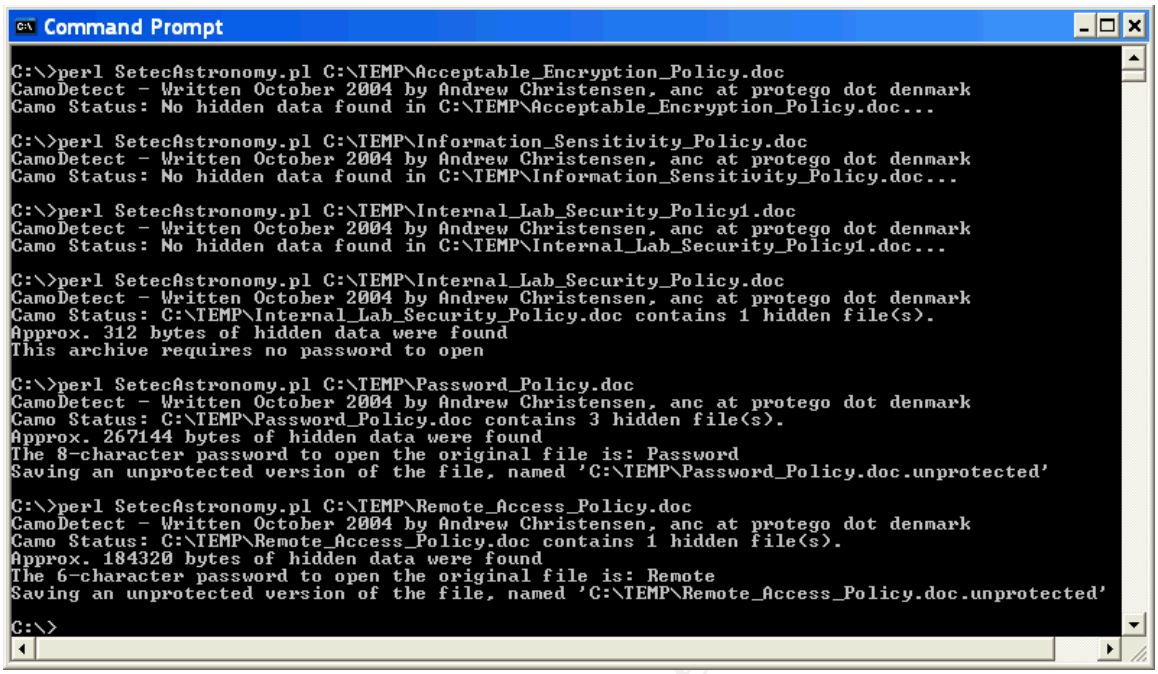


Figure 18

As the output in Figure 6 shows the file named 'Password_Policy.doc' contained three hidden documents and the password to retrieve them was 'Password'. We can also see from the output that the file named 'Remote_Access_Policy.doc' contains one hidden file, which can be retrieved using the password 'Remote'.

With this new information I successfully opened the two password protected Camouflaged documents. Hidden within 'Password_Policy.doc' were three files named 'PEM-fuel-cell-large.jpg', 'Hydrocarbon%20fuel%20cell%20page2.jpg', and 'pem_fuelcell.gif'. Hidden within the 'Remote_Access_Policy.doc' was a file titled 'CAT.mdb'. I extracted these files and immediately ran the tool 'mac_daddy.pl'[13] to try and determine when the files were created, modified, and accessed. The tool 'mac_daddy.pl' was used to gather MAC times (modified, accessed, changed times) much like I did on the original images using 'fls'. I used 'mac_daddy.pl' to gether these times because the files are visible to the operating system through normal means. 'mac_daddy.pl' pulls the times from the files in a directory whereas 'fls' was used to extract the times from a raw image file.

Some of the times were changed to reflect the time that I extracted the files from their host documents. Notice that the MAC times shown for document files are different then the times listed for them earlier, this has to do with the fact that 'fls' has a parameter for adjusting the time based on the time zone the evidence originated from where 'mac_daddy.pl' does not. For this reason I used these times only to get an idea of how the files were used and related to each

[13] mac_daddy.pl is a perl script authored by Rob Lee

other time wise. A fully adjusted time line was created and is discussed later in this paper. The relevant results are shown below, with the times related to my extraction process omitted:

```
Apr 22 2004 18:31:06   33423 m.. -r-------- root     root     /mnt/winxp/TEMP/Internal_Lab_Security_Policy.doc
                       32256 m.. -r-------- root     root     /mnt/winxp/TEMP/Internal_Lab_Security_Policy1.doc
Apr 23 2004 09:15:16   30264 m.. -r-------- root     root     /mnt/winxp/TEMP/pem_fuelcell.gif
Apr 23 2004 09:21:02  208127 m.. -r-------- root     root     /mnt/winxp/TEMP/Hydrocarbon%20fuel%20cell%20page2.jpg
Apr 23 2004 09:23:23   28167 m.. -r-------- root     root     /mnt/winxp/TEMP/PEM-fuel-cell-large.jpg
Apr 23 2004 10:21:06  184320 m.. -r-------- root     root     /mnt/winxp/TEMP/CAT.mdb
Apr 23 2004 13:03:53     312 m.. -r-------- root     root     /mnt/winxp/TEMP/Opportunity.txt
Apr 23 2004 13:54:32  215895 m.. -r-------- root     root     /mnt/winxp/TEMP/Remote_Access_Policy.doc
Apr 23 2004 13:55:26  307935 m.. -r-------- root     root     /mnt/winxp/TEMP/Password_Policy.doc
Apr 23 2004 13:59:09     312 .a. -r-------- root     root     /mnt/winxp/TEMP/Opportunity.txt
Apr 23 2004 13:59:36  208127 .a. -r-------- root     root     /mnt/winxp/TEMP/Hydrocarbon%20fuel%20cell%20page2.jpg
                       28167 .a. -r-------- root     root     /mnt/winxp/TEMP/PEM-fuel-cell-large.jpg
                       30264 .a. -r-------- root     root     /mnt/winxp/TEMP/pem_fuelcell.gif
Apr 23 2004 14:00:14  184320 .a. -r-------- root     root     /mnt/winxp/TEMP/CAT.mdb
Apr 23 2004 16:10:50   22528 m.. -r-------- root     root     /mnt/winxp/TEMP/Acceptable_Encryption_Policy.doc
Apr 23 2004 16:11:10   42496 m.. -r-------- root     root     /mnt/winxp/TEMP/Information_Sensitivity_Policy.doc
```

Figure 19

As a test to better understand how MAC times related to the use of Camouflage I hide a test file in a test document. I then extracted it to a new file and used 'mac_daddy.pl' again to gather a timeline of the test files used. The test showed that the Modified time of the recovered/hidden file was unaffected by the camouflaging and un-camouflaging processes. The test also seemed to indicate that the Accessed time of the recovered file is the time that the file was accessed by Camouflage in order to hide it and is very close to the Modified time of the host document used. The test also showed that the Accessed times of the host document and the Created/Changed time for all documents is the time which the hidden files were recovered using Camouflage. With this test information it is safe to assume that the Modified and Accessed times for recovered files and the Modified times for host documents are relevant in timeline analysis. The remaining times are affected by the investigative process and should not be used for timeline construction.

Generating a Timeline

With this new information about the MAC times of the hidden files I decided to create a compete timeline incorporating the information on the files from the original floppy disk image as well as the related information from the camouflaged files. Since mac_daddy doesn't have a way to adjust the times based on time zone I decided to gather all the MAC times without this sort of adjustment so that all the times would be accurate relative to each other. I ran the command 'fls' again against the floppy disk image, this time telling it to output the data in the 'mactime' format. I then ran the command 'mac-robber'

against the directory that contained the recovered camouflaged files. 'mac-robber' is a tool from The Sleuth Kit that gather MAC information on the files in a directory and outputs that information in 'mactime' format. I then used the Linux tool 'cat' to merge the two sets of MAC time information into one file. This file was then sorted and formatted into a timeline using the 'mactime' from The Sleuth Kit. This output is formatted the same as the 'mac_daddy.pl' output we saw earlier.

The timeline seemed to indicate that the hidden files were all hidden the morning of Friday April 23, 2004. It appears that the Camouflage file, CamShell.dll was deleted on Monday April 26, 2004 the morning of the day that the floppy disk was seized from Mr. Leszczynski. The resulting expanded timeline is shown below:

```
Sat Feb 03 2001 19:44:16    36864 m.. -/-rwxrwxrwx 0   0   5        /CamShell.dll (_AMSHELL.DLL) (deleted)
Thu Apr 22 2004 16:31:06    33423 m.. -/-rwxrwxrwx 0   0   17       /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                            32256 m.. -/-rwxrwxrwx 0   0   13       /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Fri Apr 23 2004 08:15:17    30264 ma. -rwxr--r--   99  99  2431695  /images/pem_fuelcell.gif
Fri Apr 23 2004 08:21:02   208127 ma. -rwxr--r--   99  99  2431694  /images/Hydrocarbon fuel cell page2.jpg
Fri Apr 23 2004 08:23:24    28167 ma. -rwxr--r--   99  99  2431693  /images/PEM-fuel-cell-large.jpg
Fri Apr 23 2004 09:21:06   184320 ma. -rwxr--r--   99  99  2431696  /images/CAT.mdb
Fri Apr 23 2004 10:53:56      727 m.. -/-rwxrwxrwx 0   0   28       /_ndex.htm (deleted)
Fri Apr 23 2004 11:54:32   215895 m.. -/-rwxrwxrwx 0   0   23       /Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26   307935 m.. -/-rwxrwxrwx 0   0   20       /Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 12:03:53      312 ma. -rwxr--r--   99  99  2431692  /images/Opportunity.txt
Fri Apr 23 2004 14:10:50    22528 m.. -/-rwxrwxrwx 0   0   27       /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10    42496 m.. -/-rwxrwxrwx 0   0   9        /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Sun Apr 25 2004 00:00:00        0 .a. -/-rwxrwxrwx 0   0   3        /RJL         (Volume Label Entry)
Sun Apr 25 2004 10:53:40        0 m.c -/-rwxrwxrwx 0   0   3        /RJL         (Volume Label Entry)
Mon Apr 26 2004 00:00:00    42496 .a. -/-rwxrwxrwx 0   0   9        /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
                            36864 .a. -/-rwxrwxrwx 0   0   5        /CamShell.dll (_AMSHELL.DLL) (deleted)
                            22528 .a. -/-rwxrwxrwx 0   0   27       /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
                            32256 .a. -/-rwxrwxrwx 0   0   13       /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
                            33423 .a. -/-rwxrwxrwx 0   0   17       /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                           215895 .a. -/-rwxrwxrwx 0   0   23       /Remote_Access_Policy.doc (REMOTE~1.DOC)
                              727 .a. -/-rwxrwxrwx 0   0   28       /_ndex.htm (deleted)
                           307935 .a. -/-rwxrwxrwx 0   0   20       /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:18    36864 ..c -/-rwxrwxrwx 0   0   5        /CamShell.dll (_AMSHELL.DLL) (deleted)
Mon Apr 26 2004 09:46:20    42496 ..c -/-rwxrwxrwx 0   0   9        /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Mon Apr 26 2004 09:46:22    32256 ..c -/-rwxrwxrwx 0   0   13       /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Mon Apr 26 2004 09:46:24    33423 ..c -/-rwxrwxrwx 0   0   17       /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
Mon Apr 26 2004 09:46:26   307935 ..c -/-rwxrwxrwx 0   0   20       /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36   215895 ..c -/-rwxrwxrwx 0   0   23       /Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44    22528 ..c -/-rwxrwxrwx 0   0   27       /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36      727 ..c -/-rwxrwxrwx 0   0   28       /_ndex.htm (deleted)
```

Figure 20

Analyzing the Hidden Files

After having gather the MAC time information from the recovered file I ran the Linux command 'file' against them to see if it reported them to be the same as their file extensions suggested they were. The results from the 'file' command are below:

```
[root@Daisy recovered_files]# file PEM-fuel-cell-large.jpg
PEM-fuel-cell-large.jpg: JPEG image data, JFIF standard 1.02

[root@Daisy recovered_files]# file Hydrocarbon%20fuel%20cell%20page2.jpg
Hydrocarbon%20fuel%20cell%20page2.jpg: JPEG image data, JFIF standard 1.02

[root@Daisy recovered_files]# file pem_fuelcell.gif
pem_fuelcell.gif: GIF image data, version 89a, 550 x 373

[root@Daisy recovered_files]# file CAT.mdb
CAT.mdb: Microsoft Access Database
```

Figure 21

I then opened each of the image files with Microsoft® Windows Picture and Fax Viewer which came installed with my default installation of Windows XP and the database file was opened with Microsoft® Access. What was contained in those files is shown below and much of it appeared to be confidential and proprietary information belonging to Ballard Industries. At this point it appeared that I may have found the proverbial "Smoking Gun". The contents of these files are shown below:

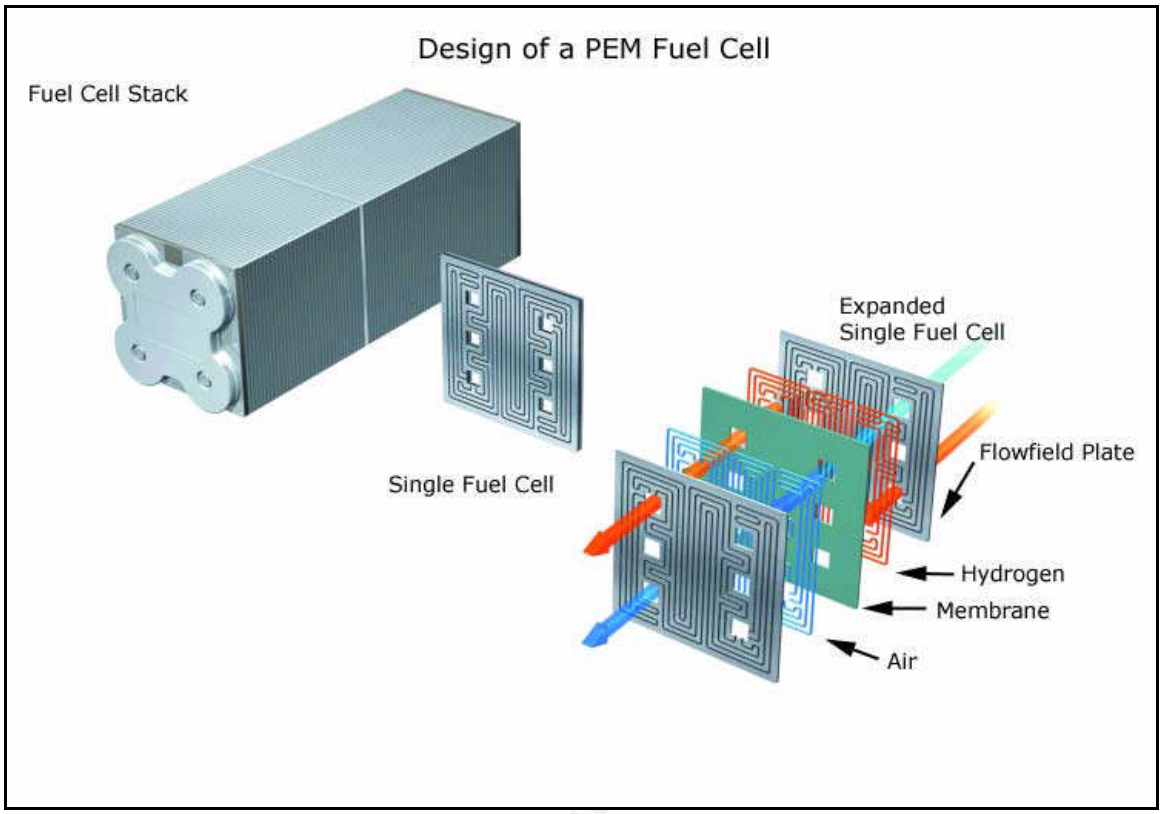


Figure 22 - PEM-fuel-cell-large.jpg

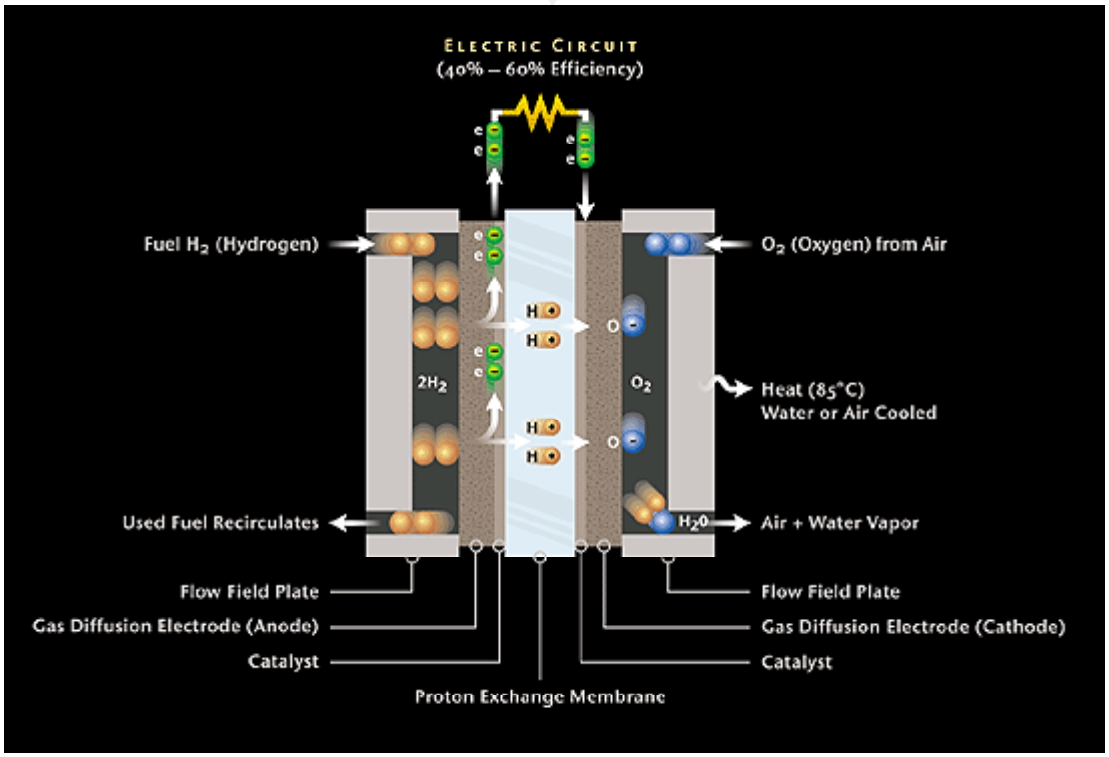


Figure 23 - pem_fuelcell.gif

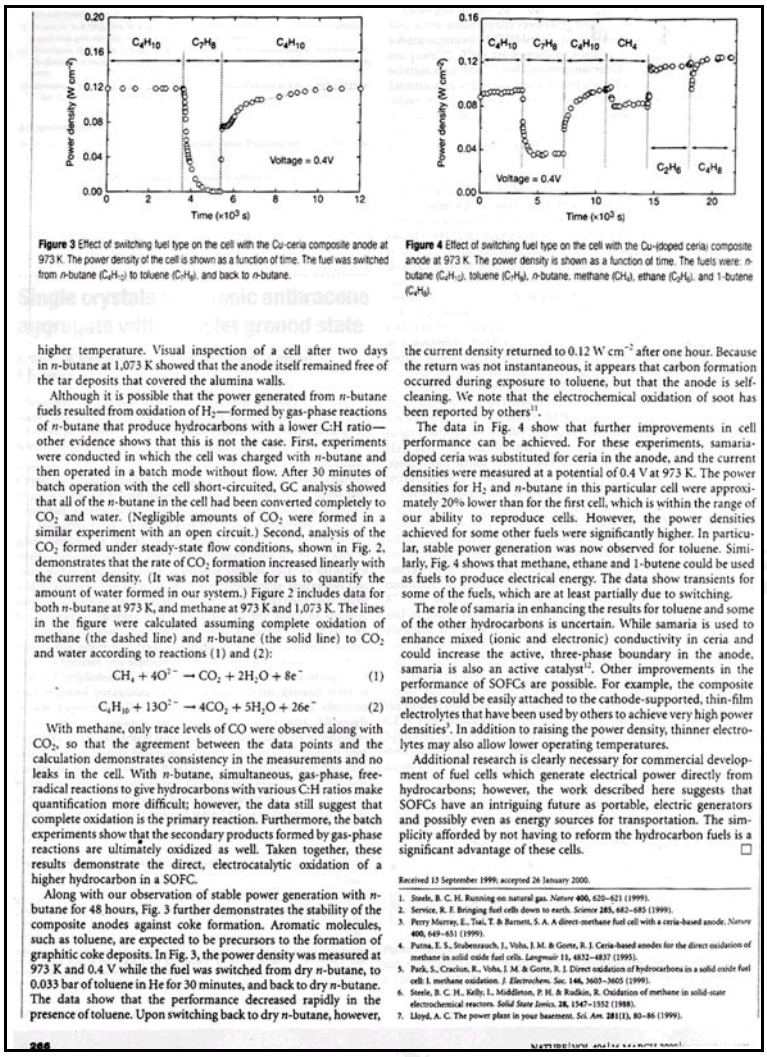


Figure 3 Effect of switching fuel type on the cell with the Cu-ceria composite anode at 973 K. The power density of the cell is shown as a function of time. The fuel was switched from *n*-butane (C_4H_{10}) to toluene (C_7H_8), and back to *n*-butane.

Figure 4 Effect of switching fuel type on the cell with the Cu-Idoped 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):

$$CH_4 + 4O^{2-} \rightarrow CO_2 + 2H_2O + 8e^- \quad (1)$$

$$C_4H_{10} + 13O^{2-} \rightarrow 4CO_2 + 5H_2O + 26e^- \quad (2)$$

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 others[11].

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.

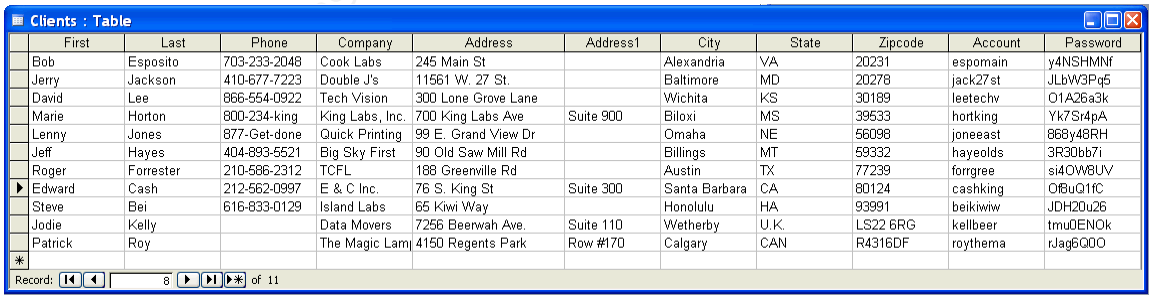
The role of samaria in enhancing the results for toluene and some of the other hydrocarbons is uncertain. While samaria is used to enhance mixed (ionic and electronic) conductivity in ceria and could increase the active, three-phase boundary in the anode, samaria is also an active catalyst[12]. Other 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 densities[3]. 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 26 January 2000.

1. Steele, B. C. H. Running on natural gas. *Nature* **400**, 620–621 (1999).
2. Service, R. F. Bringing fuel cells down to earth. *Science* **285**, 682–685 (1999).
3. Perry Murray, E., Tsai, T. & Barnett, S. A. A direct-methane fuel cell with a ceria-based anode. *Nature* **400**, 649–651 (1999).
4. Putna, E. S., Stubenrauch, J., Vohs, J. M. & Gorte, R. J. Ceria-based anodes for the direct oxidation of methane in solid oxide fuel cells. *Langmuir* **11**, 4832–4837 (1995).
5. Park, S., Craciun, R., Vohs, J. M. & Gorte, R. J. 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., Kelly, I., Middleton, P. H. & Rudkin, R. Oxidation of methane in solid-state electrochemical reactors. *Solid State Ionics.* **28**, 1547–1552 (1988).
7. Lloyd, A. C. The power plant in your basement. *Sci. Am.* **281**(1), 80–86 (1999).

Figure 24 - Hydrocarbon%20fuel%20cell%20page2.jpg



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	JLbW3Pq5
	David	Lee	866-554-0922	Tech Vision	300 Lone Grove Lane		Wichita	KS	30189	leetechv	O1A26a3k
	Marie	Horton	800-234-king	King Labs, Inc.	700 King Labs Ave	Suite 900	Biloxi	MS	39533	hortking	Yk7Sr4pA
	Lenny	Jones	877-Get-done	Quick Printing	99 E. Grand View Dr		Omaha	NE	56098	joneeast	868y48RH
	Jeff	Hayes	404-893-5521	Big Sky First	90 Old Saw Mill Rd		Billings	MT	59332	hayeolds	3R3Obb7i
	Roger	Forrester	210-586-2312	TCFL	188 Greenville Rd		Austin	TX	77239	forrgree	si4OW8UV
▶	Edward	Cash	212-562-0997	E & C Inc.	76 S. King St	Suite 300	Santa Barbara	CA	80124	cashking	OfBuQ1fC
	Steve	Bei	616-833-0129	Island Labs	65 Kiwi Way		Honolulu	HA	93991	beikiwiw	JDH20u26
	Jodie	Kelly		Data Movers	7256 Beerwah Ave.	Suite 110	Wetherby	U.K.	LS22 6RG	kellbeer	tmu0ENOk
	Patrick	Roy		The Magic Lam	4150 Regents Park	Row #170	Calgary	CAN	R4316DF	roythema	rJag6Q0O
*											

Record: 8 of 11

Figure 25 - CAT.mdb

I took all of the results and reported them to David Keen of Ballard Industries. I also recommended a meeting with Ballard Industries legal council to discuss the legal implications of my findings.

What Does the Evidence Show?

With only the floppy disk image to examine it would be difficult to prove that Mr. Leszczynski was the one the Camouflaged the proprietary information found on the floppy. It is, however, very likely that he knew the information was hidden since his name appears in one of the recovered files along with text indicating his intent to sell Ballard Industries trade secrets. The evidence from the floppy disk could be combined with other evidence to make a strong case against Mr. Leszczynski.

Determining whether or not Mr. Leszczynski is guilty of breaking any laws is up to the courts and it is up to Ballard Industries legal council if they wish to press charges. It is likely that Mr. Leszczynski and those who received information from him are guilty of violating the Uniform Trade Secrets Act. According to IPWatchdog.com:

> "Virtually all states have adopted a portion of or modified version of the Uniform Trade Secrets Act, which was drafted by the National Conference of Commissioners on Uniform State Laws in 1970 and amended in 1985."[14]

One thing that they will need to take into consideration is whether or not Ballard Industries has taken the necessary steps to protect their proprietary information. One source says:

> "Information that qualifies as a trade secret is subject to legal protection (against theft and misappropriation) as a form of valuable property--but only if the owner has taken the necessary steps to preserve its secrecy. If the owner has not diligently tried to keep the information secret, courts will usually refuse to extend any help to the trade secret owner if others learn of the information."[15]

The only information I have regarding Ballard Industries related to this is that they have a policy against removing floppy disks from the R&D lab. If this policy and other safeguards they have in place are enough to qualify the recovered data as trade secrets they may have a case against Mr. Leszczynski. If found guilty under the Economic Espionage Act of 1996 Mr. Leszczynski could be fined up to $500,000.00 and Rift, Inc. could be fined up to $5,000,000.00.[16] This information was included in my report and given to Mr. Keen along with a print out of the recovered files and a CD-ROM of the recovered files.

---

[14] http://www.ipwatchdog.com/tradesecret.html
[15] http://www.marketingtoday.com/legal/tradesec.htm
[16] http://cobrands.business.findlaw.com/intellectual_property/nolo/faq/90781CA8-0ECE-4E38-BF9E29F7A6DA5830.html

References for Part 1

Carrier, Brian. “Overview.” The Sleuth Kit. 2 Nov. 2004. URL: http://www.sleuthkit.org/sleuthkit/index.php (4 Dec. 2004).

Christensen, Andrew. “CamoDetect” 23 Oct. 2004. URL: http://packetstormsecurity.nl/crypt/stego/camouflage/SetecAstronomy.pl (5 Dec. 2004).

Elias, Stepher. “Trade Secret Law: Overview.” 1998. URL: http://www.marketingtoday.com/legal/tradesec.htm (11 Dec. 2004).

FindLaw. “Trade Secret Basics FAQ” 2002. URL: http://cobrands.business.findlaw.com/intellectual_property/nolo/faq/90781CA8-0ECE-4E38-BF9E29F7A6DA5830.html (11 Dec. 2004).

Google. “Google Search.” URL: http://www.google.com (4 Dec. 2004).

Guillermito, “(easily) Breaking a (very weak) steganography software.” 6 May 2003. URL: http://guillermito2.net/stegano/camouflage/ (5 Dec. 2004).

IPWatchdog. “Trade Secret Law.” 24 Nov. 2004. URL: http://www.ipwatchdog.com/tradesecret.html (11 Dec. 2004).

Payne, Steve. “DataLifter Forensicware Tools.” 14 Nov. 2004. URL: http://www.datalifter.com (4 Dec. 2004).

Perkel, Marc. “Linux MAN Pages.” 9 Feb. 2004. URL: http://linux.ctyme.com/ (4 Dec. 2004).

Rivest, Ronald R. “RFC 1321 - The MD5 Message-Digest Algorithm.” Apr. 1992. URL: http://www.faqs.org/rfcs/rfc1321.html (4 Dec. 2004).

SANS Institute. “GIAC: Global Information Assurance Certification - GIAC Certified Forensic Analyst (GCFA) Practical Assignment.” 30 Apr. 2004. URL: http://www.giac.org/GCFA_assign_15.php (4 Dec. 2004).

SANS Institute. “The SANS Security Policy Project.” URL: http://www.sans.org/resources/policies/#template (4 Dec. 2004).

Sourceforge. “Foremost - Latest version 0.69.” 19 Nov. 2004. URL: http://foremost.sourceforge.net/ (4 Dec. 2004).

TranceAddict Forums. ” Camoflagued Mp3s Contain A Backdoor Beware.” 12 Dec. 2002. URL: http://www.tranceaddict.com/forums/archive/topic/79627-1.html (4 Dec. 2004).

Unfiction. “Download Camouflage” 26 Mar. 2003. URL: http://camouflage.unfiction.com/ (5 Dec. 2004).

Part 2 – Perform Forensic Tool Validation

Scope

When making an image of a disk or other electronic evidence, it is recommended by many sources[17][18] that the examination or working copy be saved to media that has been made forensically sterile.

Forensically sterile media is free from any residual data left over from previous uses of the media or the manufacturing process. Typically a series of 'null' values are written on every sector of the disk to eliminate any previous data.

In this test we will take a hard drive that contains data and make it forensically sterile. We will then do a series of test to verify that it has been properly prepared.

Tool Description

The tool that will be used to make the hard drive forensically sterile is called Sterilize and it available free of charge from the CyberSecurity® Institute[19].

The read me file that was downloaded with the Sterilize tool has a very good explanation as to why using sterile media is a good practice when doing computer forensics. From the Readme.txt file:

> "Why use Sterilize?
>
> It is extremely important to remove any residual data (or the possibility that data exists in some form) from the media to be used as working copies for a forensic examination. This sterilization process should be documented and visually verified by the forensic examiner, leaving no doubt that whatever is found on the working copy during a forensic examination is/was also present on the original media.
>
> A true bit stream or forensic imaging process by itself should overwrite any existing data on the working copy media. When the original media and working media hash the same, the examiner can say that a forensically sound image has been created. The extra step of sanitizing media prior to creating forensic working copies however will help the forensic examiner in dealing with possible "junk science" attacks, and will result in one less aspect of the overall procedures and methodology used

[17] http://www.itsecurity.com/papers/halcrow1.htm
[18] http://www.pimall.com/nais/nl/ecomputerf.html
[19] http://www.cybersecurityinstitute.biz/software/

being brought under scrutiny."[20]

The version used in this test is 1.02.7 by Robert Orr. This version will wipe the media so that it is forensically sterile. It will also wipe the data in such a way as to be compliant with several standards for sanitizing media including the Department of Defense standards found in DOD 5220.22-M[21] Chapter 8. According to the Readme file, Sterilize will write the hexadecimal value '0x00' to every sector on the disk. It will then check random sectors on the drive to verify that they do in fact contain '0x00'. The Sterilize program also provide the capability to do a 128-bit checksum on the disk as well as manually checking sectors on the disk to verify that the value '0x00' has been written to them. Sterilize also has a very nice feature in that all the results of these actions can be written to a report file.

In order to use Sterilize I first created a Windows boot floppy. It is recommended that this boot floppy be made into a forensic boot floppy using the program 'mod_com.exe' by Dan Mares[22]. This tool prepares the floppy disk in such a way as too prevent it from accessing the system hard drive when booting the system. Once a bootable floppy is created the only other file needed is the Sterilize executable itself, 'sterliz.exe'.

Test Apparatus and Environmental Conditions

The test machine used was an old PII 333MHZ with 128MB RAM and with a Western Digital Caviar 2850 853.6 MB hard drive. The machine was not connected to a network and no operating system was loaded on the hard drive. The hard drive was formerly a data drive from another system. The hard drive contained some data files that were no longer needed but were still on the hard drive and could therefore be used to verify that the sterilization processed worked. The lab was in a locked room and all systems, media, and results were in my control or locked up until the testing was completed to ensure that no outside forces could alter the results.

Description of the Procedures

Preparation of the test system was comprised of verifying that the device boot order was floppy disk and CD-ROM before hard drive. I also verified that all the programs being used for the test ran properly.

Test results generated by Sterilize will be saved to the Sterilize boot floppy as text files following the default naming convention defined by the program.

[20] Readme.txt from the 'sterliz.zip' file downloaded from the CyberSecurity® Institute.
[21] http://www.dtic.mil/whs/directives/corres/html/522022m.htm
[22] http://www.maresware.com/maresware/freesoftware.htm

Results from verification programs will be saved as a text file with the name of the command being run followed by the number of the run in the series (i.e. command1.txt). All results will then be compiled into this document.

The test results will be verified using two basic tools. Norton DiskEdit from Symantec[23] will be used to examine the raw sector data and verify that what Sterilize shows is the same as what DiskEdit shows. The live bootable Linux CD distribution called Helix[24] will be used to verify checksum values as reported by Sterilize. Both of these tools are used from bootable media and mount the hard drive in read only mode by default. This will ensure that these verification tools will not inadvertently alter the data on the hard drive there by protecting the results of the actions taken by Sterilize.

The first series of tests will be run on the hard drive with the old data still intact. These tests will be comprised of using Sterilize to do a checksum on the data and view the data to verify that something still exists on the disk. The results will then be verified using DiskEdit and Helix. It is expected that the initial checksum created by Sterilize will not match that of the 'sum' command used in Helix. This is because the Sterilize checksum is a 128-bit checksum while the 'sum' command returns a 16-bit checksum.

The next set of tests will start with using Sterilize to wipe all the data from the hard drive. Next the same tests and verifications ran is the first set of tests will be run again. This time the tests will verify whether or not Sterilize has in fact wiped the hard drive clean and is reporting correctly that the drive has been wiped. The checksum validation using 'sum' will verify that the Sterilize checksum is valid for this set of testing. This is because the 16-bit checksum provided by 'sum' and the 128-bit checksum provided by Sterilize will both be all zero's if they are computing the checksums correctly.

Finally, if the disk has been properly cleaned, data will be placed on the disk and the two sets of testing will be repeated to verify the results from the first round of tests.

Criteria for Approval

There are three criteria for approval in this set of tests. The first is that Sterilize will accurately display the physical sectors of a hard drive. It is expected that whatever values displayed by Sterilize will be the same as the values at the same physical sectors when displayed by DiskEdit. The second criteria is that the checksum value of the hard drive as calculated by Sterilize will be the same value as calculated by the Linux command 'sum' run under Helix. The third and final criteria for approval is the visual inspection of physical sectors

[23] http://www.symantec.com/sabu/sysworks/pro/
[24] http://www.e-fense.com/helix/

on the disk using the Sterilize viewing option and the tool DiskEdit.

To gather the initial set of test data the system is booted with the Sterilize floppy. Sterilize can be run with a Graphical User Interface menu or with command line options. For these tests I used the menu system since this allowed me to easily save multiple test results to the same report file. From the Sterilize menu I first ran the checksum option. These results were saved in the Sterilize report file named '12190400.RPT'. When this had completed I ran the option to physically view the disk sectors. Since Sterilize doesn't have the ability (at least not that I could find) to print any of the viewed sectors I wrote some notes on paper indicating what was displayed. With these results in hand I booted the machine to the DiskEdit floppy. Using DiskEdit I viewed the same sectors as viewed with Sterilize. I was unable to find a way to copy the view of the data on the physical disk sectors with DiskEdit either so I again took notes on paper to compare what Sterilize showed with what DiskEdit showed. Next I booted the system to the Helix CD and ran the tool 'sum' against the hard drive. These results were saved as 'sum1.txt'.

The next set of test started by booting to the Sterilize floppy and launching the menu. From the menu I selected the 'Sterilize' option. This option warns the user that this will destroy all data on the disk and asks for conformation before continuing. I confirmed to continue and Sterilize proceeded to begin writing the hex value '0x00' to every sector of the disk. When the sterilization process had completed I ran the checksum option from the menu. These results were saved in the report file named '12190401.RPT'. I then selected the view sectors option from the Sterilize menu and viewed several disk sectors, noting on my note paper which sector were examined. Next I booted to the DiskEdit floppy and viewed the same sectors as viewed with Sterilize. After viewing the sectors I booted to the Helix CD and ran the 'sum' command again against the hard drive saving the results in the file 'sum2.txt'.

After the first two sets of tests were run data was copied to the hard disk and the tests were all run again. This time the Sterilize reports were saved as '12200400.RPT' and '12200401.RPT' and the 'sum' results were saved as 'sum3.txt' and 'sum4.txt'.

Data and Results

```
12-19-2004 02:18 : CSI Sterliz.exe report file

12-19-2004 02:18 : Report user name: Steve

12-19-2004 02:18 : **********
12-19-2004 02:18 : Start of report segment
12-19-2004 02:18 : Checksum of HD0
12-19-2004 02:18 : ----------
12-19-2004 02:18 : Drive information for HD0  (Physical Drive 80)
12-19-2004 02:18 : Physical Cylinders    : 827
12-19-2004 02:18 : Physical Heads        : 32
12-19-2004 02:18 : Physical Sectors/Track: 63
12-19-2004 02:18 : Bytes per Sector      : 512
12-19-2004 02:18 : Total Sectors         : 1667232
12-19-2004 02:18 : Total MBytes          : 814Mb
12-19-2004 02:18 : ----------
12-19-2004 02:32 : Media is NOT sterile!
12-19-2004 02:32 : Checksum of HD0  completed
12-19-2004 02:32 : --------Segment Summary--------
12-19-2004 02:32 : 128 bit checksum  : 159226a4fc74ca6aacc6a830cfb78484
12-19-2004 02:32 : Total Read Errors : 0
12-19-2004 02:32 : Total Write Errors: 0
12-19-2004 02:32 :  End of report segment
12-19-2004 02:32 : **********
12-19-2004 02:32 : Start of report segment
12-19-2004 02:32 : Sectors on HD0
12-19-2004 02:32 : ----------
12-19-2004 02:32 : Drive information for HD0  (Physical Drive 80)
12-19-2004 02:32 : Physical Cylinders    : 827
12-19-2004 02:32 : Physical Heads        : 32
12-19-2004 02:32 : Physical Sectors/Track: 63
12-19-2004 02:32 : Bytes per Sector      : 512
12-19-2004 02:32 : Total Sectors         : 1667232
12-19-2004 02:32 : Total MBytes          : 814Mb
12-19-2004 02:32 : ----------
12-19-2004 02:32 :  Displayed sector 0
12-19-2004 02:32 :  Displayed sector 1
12-19-2004 02:32 :  Displayed sector 2
12-19-2004 02:32 :  Displayed sector 3
12-19-2004 02:32 : --------Segment Summary--------
12-19-2004 02:32 : Total Read Errors : 0
12-19-2004 02:32 : Total Write Errors: 0
12-19-2004 02:32 :  End of report segment
12-19-2004 02:32 :
**********
Report generated by
        sterliz 1.02.7 (c) 2004 CyberSecurity Institute
         http://www.cybersecurityinstitute.biz
```

Figure 26 - 12190400.RPT

```
12-19-2004 22:00 : CSI Sterliz.exe report file

12-19-2004 22:00 : Report user name: Steve

12-19-2004 22:00 : **********
12-19-2004 22:00 : Start of report segment
12-19-2004 22:00 : Sterilize drive HD0 ...
12-19-2004 22:00 : ----------
12-19-2004 22:00 : Drive information for HD0  (Physical Drive 80)
12-19-2004 22:00 : Physical Cylinders    : 827
12-19-2004 22:00 : Physical Heads        : 32
12-19-2004 22:00 : Physical Sectors/Track: 63
12-19-2004 22:00 : Bytes per Sector      : 512
12-19-2004 22:00 : Total Sectors         : 1667232
12-19-2004 22:00 : Total MBytes          : 814Mb
12-19-2004 22:00 : ----------
12-19-2004 22:00 : Sterilize start
12-19-2004 22:00 : Writing 00 bytes to selected drive
12-19-2004 22:07 : Random 5000 Sector Verify of Drive: HD0
12-19-2004 22:08 : Random selection of 5000 sectors verified
12-19-2004 22:08 : Sterilize complete for HD0
12-19-2004 22:08 : 0 Read Errors
12-19-2004 22:08 : 0 Write Errors
12-19-2004 22:21 : Sterilize of HD0  completed
12-19-2004 22:21 : --------Segment Summary--------
12-19-2004 22:21 : Total Read Errors : 0
12-19-2004 22:21 : Total Write Errors: 0
12-19-2004 22:21 :  End of report segment
12-19-2004 22:21 : **********
12-19-2004 22:21 : Start of report segment
12-19-2004 22:21 : Checksum of HD0
12-19-2004 22:21 : ----------
12-19-2004 22:21 : Drive information for HD0  (Physical Drive 80)
12-19-2004 22:21 : Physical Cylinders    : 827
12-19-2004 22:21 : Physical Heads        : 32
12-19-2004 22:21 : Physical Sectors/Track: 63
12-19-2004 22:21 : Bytes per Sector      : 512
12-19-2004 22:21 : Total Sectors         : 1667232
12-19-2004 22:21 : Total MBytes          : 814Mb
12-19-2004 22:21 : ----------
12-19-2004 22:39 : Checksum of HD0  completed
12-19-2004 22:39 : --------Segment Summary--------
12-19-2004 22:39 : 128 bit checksum  : 00000000000000000000000000000000
12-19-2004 22:39 : Total Read Errors : 0
12-19-2004 22:39 : Total Write Errors: 0
12-19-2004 22:39 :  End of report segment
12-19-2004 22:39 : **********
12-19-2004 22:39 : Start of report segment
12-19-2004 22:39 : Sectors on HD0
12-19-2004 22:39 : ----------
12-19-2004 22:39 : Drive information for HD0  (Physical Drive 80)
12-19-2004 22:39 : Physical Cylinders    : 827
12-19-2004 22:39 : Physical Heads        : 32
12-19-2004 22:39 : Physical Sectors/Track: 63
12-19-2004 22:39 : Bytes per Sector      : 512
12-19-2004 22:39 : Total Sectors         : 1667232
12-19-2004 22:39 : Total MBytes          : 814Mb
12-19-2004 22:39 : ----------
12-19-2004 22:39 :  Displayed sector 0
12-19-2004 22:39 :  Displayed sector 1
12-19-2004 22:39 :  Displayed sector 2
12-19-2004 22:39 :  Displayed sector 3
12-19-2004 22:39 :  Displayed sector 4
12-19-2004 22:39 :  Displayed sector 5
12-19-2004 22:39 :  Displayed sector 6
12-19-2004 22:40 :  Displayed sector 1667000
12-19-2004 22:40 :  Displayed sector 1667001
12-19-2004 22:40 :  Displayed sector 1667002
12-19-2004 22:40 :  Displayed sector 1667003
12-19-2004 22:40 :  Displayed sector 1667230
12-19-2004 22:40 :  Displayed sector 1667231
12-19-2004 22:40 :  Displayed sector 0
12-19-2004 22:40 :  Displayed sector 1
12-19-2004 22:40 :  Displayed sector 0
12-19-2004 22:40 :  Displayed sector 1667231
```

Figure 27 - 12190401.RPT

```
12-20-2004 00:29 : CSI Sterliz.exe report file

12-20-2004 00:29 : Report user name: Steve

12-20-2004 00:29 : ********************
12-20-2004 00:29 : Start of report segment
12-20-2004 00:30 : Checksum of HD0
12-20-2004 00:30 : ----------------------------------------------------
12-20-2004 00:30 : Drive information for HD0  (Physical Drive 80)
12-20-2004 00:30 : Physical Cylinders    : 827
12-20-2004 00:30 : Physical Heads        : 32
12-20-2004 00:30 : Physical Sectors/Track: 63
12-20-2004 00:30 : Bytes per Sector      : 512
12-20-2004 00:30 : Total Sectors         : 1667232
12-20-2004 00:30 : Total MBytes          : 814Mb
12-20-2004 00:30 : ----------------------------------------------------
12-20-2004 00:37 : Media is NOT sterile!
12-20-2004 00:37 : Checksum of HD0  completed
12-20-2004 00:37 : --------Segment Summary--------
12-20-2004 00:37 : 128 bit checksum  : c373a372dbafb51220e122286223b1a9
12-20-2004 00:37 : Total Read Errors : 0
12-20-2004 00:37 : Total Write Errors: 0
12-20-2004 00:37 :  End of report segment
12-20-2004 00:38 : ********************
12-20-2004 00:38 : Start of report segment
12-20-2004 00:38 : Sectors on HD0
12-20-2004 00:38 : ----------------------------------------------------
12-20-2004 00:38 : Drive information for HD0  (Physical Drive 80)
12-20-2004 00:38 : Physical Cylinders    : 827
12-20-2004 00:38 : Physical Heads        : 32
12-20-2004 00:38 : Physical Sectors/Track: 63
12-20-2004 00:38 : Bytes per Sector      : 512
12-20-2004 00:38 : Total Sectors         : 1667232
12-20-2004 00:38 : Total MBytes          : 814Mb
12-20-2004 00:38 : ----------------------------------------------------
12-20-2004 00:38 :  Displayed sector 0
12-20-2004 00:38 :  Displayed sector 1
12-20-2004 00:38 :  Displayed sector 2
12-20-2004 00:39 :  Displayed sector 3
12-20-2004 00:39 :  Displayed sector 4
12-20-2004 00:39 :  Displayed sector 5
12-20-2004 00:39 :  Displayed sector 6
12-20-2004 00:39 :  Displayed sector 7
12-20-2004 00:39 :  Displayed sector 8
12-20-2004 00:39 :  Displayed sector 9
12-20-2004 00:39 :  Displayed sector 10
12-20-2004 00:39 :  Displayed sector 62
12-20-2004 00:39 :  Displayed sector 63
12-20-2004 00:39 :  Displayed sector 64
12-20-2004 00:39 :  Displayed sector 65
12-20-2004 00:39 :  Displayed sector 66
12-20-2004 00:39 :  Displayed sector 67
12-20-2004 00:39 : --------Segment Summary--------
12-20-2004 00:39 : Total Read Errors : 0
12-20-2004 00:39 : Total Write Errors: 0
12-20-2004 00:39 :  End of report segment
12-20-2004 00:39 :
**********************
Report generated by
            sterliz 1.02.7 (c) 2004 CyberSecurity Institute
             http://www.cybersecurityinstitute.biz

Report generated by
            sterliz 1.02.7 (c) 2004 CyberSecurity Institute
             http://www.cybersecurityinstitute.biz
```

Figure 28 - 12200400.RPT

```
12-20-2004 01:01 : CSI Sterliz.exe report file

12-20-2004 01:01 : Report user name: Steve

12-20-2004 01:01 : ********************
12-20-2004 01:01 : Start of report segment
12-20-2004 01:01 : Sterilize drive HD0 ...
12-20-2004 01:01 : ----------------------------------------------------
12-20-2004 01:01 : Drive information for HD0  (Physical Drive 80)
12-20-2004 01:01 : Physical Cylinders    : 827
12-20-2004 01:01 : Physical Heads        : 32
12-20-2004 01:01 : Physical Sectors/Track: 63
12-20-2004 01:01 : Bytes per Sector      : 512
12-20-2004 01:01 : Total Sectors         : 1667232
12-20-2004 01:01 : Total MBytes          : 814Mb
12-20-2004 01:01 : ----------------------------------------------------
12-20-2004 01:02 : Sterilize start
12-20-2004 01:02 : Writing 00 bytes to selected drive
12-20-2004 01:08 : Random 5000 Sector Verify of Drive: HD0
12-20-2004 01:10 : Random selection of 5000 sectors verified
12-20-2004 01:10 : Sterilize complete for HD0
12-20-2004 01:10 : 0 Read Errors
12-20-2004 01:10 : 0 Write Errors
12-20-2004 01:10 : Sterilize of HD0  completed
12-20-2004 01:10 : --------Segment Summary--------
12-20-2004 01:10 : Total Read Errors : 0
12-20-2004 01:10 : Total Write Errors: 0
12-20-2004 01:10 :  End of report segment
12-20-2004 01:10 : ********************
12-20-2004 01:10 : Start of report segment
12-20-2004 01:10 : Checksum of HD0
12-20-2004 01:10 : ----------------------------------------------------
12-20-2004 01:10 : Drive information for HD0  (Physical Drive 80)
12-20-2004 01:10 : Physical Cylinders    : 827
12-20-2004 01:10 : Physical Heads        : 32
12-20-2004 01:10 : Physical Sectors/Track: 63
12-20-2004 01:10 : Bytes per Sector      : 512
12-20-2004 01:10 : Total Sectors         : 1667232
12-20-2004 01:10 : Total MBytes          : 814Mb
12-20-2004 01:10 : ----------------------------------------------------
12-20-2004 01:18 : Checksum of HD0  completed
12-20-2004 01:18 : --------Segment Summary--------
12-20-2004 01:18 : 128 bit checksum  : 00000000000000000000000000000000
12-20-2004 01:18 : Total Read Errors : 0
12-20-2004 01:18 : Total Write Errors: 0
12-20-2004 01:18 :  End of report segment
12-20-2004 01:19 : ********************
12-20-2004 01:19 : Start of report segment
12-20-2004 01:19 : Sectors on HD0
12-20-2004 01:19 : ----------------------------------------------------
12-20-2004 01:19 : Drive information for HD0  (Physical Drive 80)
12-20-2004 01:19 : Physical Cylinders    : 827
12-20-2004 01:19 : Physical Heads        : 32
12-20-2004 01:19 : Physical Sectors/Track: 63
12-20-2004 01:19 : Bytes per Sector      : 512
12-20-2004 01:19 : Total Sectors         : 1667232
12-20-2004 01:19 : Total MBytes          : 814Mb
12-20-2004 01:19 : ----------------------------------------------------
12-20-2004 01:19 :  Displayed sector 0
12-20-2004 01:19 :  Displayed sector 1
12-20-2004 01:19 :  Displayed sector 2
12-20-2004 01:19 :  Displayed sector 3
12-20-2004 01:19 :  Displayed sector 62
12-20-2004 01:19 :  Displayed sector 63
12-20-2004 01:19 :  Displayed sector 64
12-20-2004 01:19 :  Displayed sector 65

Report generated by
            sterliz 1.02.7 (c) 2004 CyberSecurity Institute
             http://www.cybersecurityinstitute.biz
```

Figure 29 - 12200401.RPT

```
18922 833616
```

Figure 30 - sum1.txt

```
00000 833616
```

Figure 31 - sum2.txt

```
59575 833616
```

Figure 32 - sum3.txt

```
00000 833616
```

Figure 33 - sum4.txt

The results were as expected and all criteria for approval were met. By comparing the results on the report files and the results from manually viewing the disk sectors we see that Sterilize does exactly what it claims to do. Sterilizing a disk overwrites all existing data with a series of hex value '0x00' which we verified using the 'sum' command as well as visually inspecting the disk sectors using DiskEdit. Sterilize also provide an accurate view of the physical sectors which we verified by comparing the results from the Sterilize viewer with the results from DiskEdit. It was also verified to a lesser degree that the checksum feature appears to be accurate, at least when calculating the checksum on a sterilized drive. This was verified using the 'sum' command and verifying that both it and Sterilize returned a zero value when calculating the checksum on the sterilized drive.

Analysis

Thanks to the friendly report format used by the Sterilize reports it would be very easy for an investigator to see whether or not a disk has been wiped clean and made forensically sterile. An investigator could also use Sterilize to access the physical sector of a disk. This would allow the investigator to view data that my otherwise be obfuscated by the operating system or by nefarious means.

Presentation

Using the reports generated by Sterilize it would be very easy for an investigator to show that the media had been made forensically sterile before being used. The reports would document when the sterilization occurred and that the sterilization process had been verified.

The reports are organized very well and would make explaining them to others quite easy. Each option within Sterilize generates its own report segment. These report segments show the time that each activity took place,

what that activity was, and the results of that action. When the checksum for the test disk was generated the following report entry was made:

```
12-19-2004 02:18 : ********************
12-19-2004 02:18 : Start of report segment
12-19-2004 02:18 : Checksum of HD0
12-19-2004 02:18 : ------------------------------------------------------
12-19-2004 02:18 : Drive information for HD0  (Physical Drive 80)
12-19-2004 02:18 : Physical Cylinders    : 827
12-19-2004 02:18 : Physical Heads        : 32
12-19-2004 02:18 : Physical Sectors/Track: 63
12-19-2004 02:18 : Bytes per Sector      : 512
12-19-2004 02:18 : Total Sectors         : 1667232
12-19-2004 02:18 : Total MBytes          : 814Mb
12-19-2004 02:18 : ------------------------------------------------------
12-19-2004 02:32 : Media is NOT sterile!
12-19-2004 02:32 : Checksum of HD0  completed
12-19-2004 02:32 : --------Segment Summary--------
12-19-2004 02:32 : 128 bit checksum  : 159226a4fc74ca6aacc6a830cfb78484
12-19-2004 02:32 : Total Read Errors : 0
12-19-2004 02:32 : Total Write Errors: 0
12-19-2004 02:32 :  End of report segment
```

This shows that at 02:18 on Dec. 19, 2004 a checksum was started on the primary hard drive (HD0). It then shows us the drive information for the disk being analyzed. We see that Sterilize detected that the disk contained data and was not sterile followed by the checksum for the data found on the drive. Finally we see that during the checksum process Sterilize encountered no read/write errors indicating that the media is sound and that the end of that report segment has been reached. All the options create similar entries in the report and are equally easy to explain to the court or others.

These reports would be easy to admit to the court as process verification in their native form. The reports are easy to read and require no modification or formatting. A simple print out should be very easy for anyone involved to read and understand with very little explanation.

Conclusion

These tests were successful and show that Sterilize is a very convenient tool for creating forensically sterile media. Sterilize has added features that allow for verification that the media has been successfully sterilized and produces a very nice report of the actions and verifications.

This tool doesn't require any changes to make it more forensically sound. The only recommendation I have would be increasing functionality. The main functionality improvements are already in development for future releases such as the ability to do a MD5 hash of a disk.

Using Sterilize to make media forensically sterile should be the first step any investigator takes before imaging a disk. The investigator can rest assured that

by using this tool to prepare the media he/she will not have any residual data on the working copy and will be provided with a nice report should that fact ever be questioned.

References for Part 2

Dept. of Defense. “DoD 5220.22-M, ‘National Industrial Security Program Operating Manual’.” Jan. 1995. URL: http://www.dtic.mil/whs/directives/corres/html/522022m.htm (18 Dec. 2004)

e-Fense, Inc. “Helix.” 7 Dec. 2004. URL: http://www.e-fense.com/helix/ (18 Dec. 2004)

Grant, David. “Halcrow Group Ltd MIS Computer Forensic Procedures.” 2 Jun. 2002. URL: http://www.itsecurity.com/papers/halcrow1.htm (18 Dec. 2004).

Hailey, Steve. “Sterilize - FREE.” 14 Sep. 2004. URL: http://www.cybersecurityinstitute.biz/software/ (18 Dec. 2004)

Mares, Dan. “Free Software from Mares and Company.” 9 Aug. 2004. URL: http://www.maresware.com/maresware/freesoftware.htm (18 Dec. 2004)

Newsom, Dr. P. Dennis. “An Explanation of Computer Forensics.” Sep. 2000. URL: http://www.pimall.com/nais/nl/ecomputerf.html (18 Dec. 2004)

Symantec Corporation. “Norton Systemworks Premier Edition.” URL: http://www.symantec.com/sabu/sysworks/pro/ (18 Dec. 2004)