GIAC Certified Forensics Analyst (GCFA) Practical Assignment Version 1.5 Option 1 (April 30, 2004) for Alberto Partida

December 9, 2004

Two forensic cases: hidden company files and a USB memory stick

GIAC Certified Forensics Analyst (GCFA) Practical Assignment Version 1.5 Option 1 (April 30, 2004) for Alberto Partida

December 9, 2004

Two forensic cases: hidden company files and a USB memory stick

Abstract

This practical assignment presents two different scenarios where forensic analysis and techniques are applied. Both situations aim to answer, or at least to pose, several questions related with system forensics, investigation and response.

The first scenario analyses an unknown image. It describes the investigation steps taken to determine whether a seized floppy disk contains some kind of hidden information. It is a comprehensive study case that deals with steganography and follows the facts provided in the assignment brief about the sudden lost of market share suffered by Ballard Industries and its mysterious employee Mr. Robert Leszczynski Jr., the owner of the floppy disk.

The second scenario describes a forensic analysis performed on a system. It aims at providing a global overview of the investigation process. It focuses on the data present in a USB memory stick and it uses The Sleuth Kit through the autopsy forensic browser as the main supporting tool.

Throughout the whole assignment screenshots and command lines are inserted to facilitate the reading. The paper also includes the Internet sources consulted in its elaboration. The final aim of these pages is to demonstrate that, although forensic analysis needs to be aided by a clear methodology, it will always remain to be an art requiring very knowledgeable masters.

Index

1. Part 1- Analysis of an unknown image

1.1 Examination details

1.1.1 Obtaining the image

The next paragraphs describe, in chronological order, the detailed steps I followed to download, uncompress, verify and mount the floppy image (receiving and handling the image).

The first action I undertook was to download the floppy image from the URL provided in the GIAC GCFA practical assignment brief posted in GIAC momgate: http://www.giac.org/gcfa/v1_5.gz

I downloaded the floppy image twice (although from the same URL) and I stored it in two different locations, the first time in my forensics workstation and the second time in a removable storage device (a USB memory stick).

```
[alber@LinuxForensics sample]$ mv v1_5.gz /mnt/flash/v1_52.gz
```

1.1.2 Steps to analyse the image

After downloading the image, the first thing I did is using md5sum to create various md5 checksums using the tools provided in the SANS Track 8 CD-ROM to be certain that the tools I use have not been tampered previously.

The first checksum I performed was the ‘gzipped’ file first (extension .gz).

```
[alber@LinuxForensics response_kit]$ ./md5sum
/home/alber/FA/sample/v1_5.gz > /home/alber/FA/sample/v1_5.gz.md5
[alber@LinuxForensics sample]$ more v1_5.gz.md5
f39239ed04e7c0c1b36bcd556d213623  /home/alber/FA/sample/v1_5.gz
```

At the moment, I will only store this md5 result in the removable storage media I already mentioned before. I can always revert to this value to check whether the file I will be working on is still the one I downloaded from the SANS web site. To facilitate the quick location of the file, I keep the checksum in a file with the extension md5. This is an arbitrary (but helpful!) decision.

The next thing I do is listing verbosely the files that are compressed in the zipped file:

```
[alber@LinuxForensics sample]$ gzip -vlN v1_5.gz
method crc date time compressed uncompressed ratio uncompressed_name
defla 948edf93 Apr 26 02:45  502408  1474560  65.9% fl-260404-RJL1.img
```

This way I can already read the real name of the image file, just as it is mentioned in the practical assignment brief: *fl-260404-RJL1.img*

I proceed to decompress the file keeping its original name:

```
[alber@LinuxForensics sample]$ gzip -vdN v1_5.gz
```

And I 'ls' the directory to check now whether the image has been decompressed.

```
[alber@LinuxForensics sample]$ ls -alu
total 1952
drwxrwxr-x    2 alber    alber        4096 Nov 20 18:01 .
drwxrwxr-x    5 alber    alber        4096 Nov 20 17:58 ..
-rwxr-xr-x    1 alber    alber     1474560 Nov 20 18:00 fl-260404-RJL1.img
-rwxr-xr-x    1 alber    alber      502408 Nov 20 17:58 v1_5.gz
-rw-rw-r--    1 alber    alber          64 Nov 20 17:46 v1_5.gz.md5
```

Before mounting the disk image as a read only device (in order not to modify anything contained in the image), I again perform the md5 checksum, this time on the uncompressed disk image. And I check that the md5 checksum is right the md5 checksum provided by SANS web site.

```
[alber@LinuxForensics response_kit]$ ./md5sum /home/alber/sample/fl-260404-RJL1.img > fl-260404-RJL1.img.md5
```

I read the md5 file created from the image file and voila! It coincides with the md5 checksum provided by SANS.

```
[alber@LinuxForensics sample]$ more fl-260404-RJL1.img.md5
d7641eb4da871d980adbe4d371eda2ad  fl-260404-RJL1.img
```

1.1.3 Mounting the image read-only

Now I have to mount that image in a secure way i.e. as a read only image. By the way, by default, in a Red Hat box, only root can mount images.

```
[alber@LinuxForensics sample]$ mount -o ro,loop fl-260404-RJL1.img ./image/
mount: only root can do that
[alber@LinuxForensics sample]$ su root
Password:
[root@LinuxForensics sample]# pwd
/home/alber/FA/sample
[root@LinuxForensics sample]# mount -o ro,loop fl-260404-RJL1.img ../image/
[root@LinuxForensics FA]# cd image/
[root@LinuxForensics image]# ls
Acceptable_Encryption_Policy.doc    Internal_Lab_Security_Policy1.doc
Password_Policy.doc
```

```
Information_Sensitivity_Policy.doc  Internal_Lab_Security_Policy.doc
Remote_Access_Policy.doc
```

So, I have already installed read-only the floppy image and the first list of visible files can be seen above.
Next, the steps I take to analyse the image in my SANS-Track 8 forensic workstation (all necessary steps to create this forensic workstation can be found in Track 8 book 1, pages 158-163). The most used tool to answer the next questions have been autopsy.

1.2 Initial recognition

1.2.1 First tools initially used

Before starting with autopsy, I just tried different preliminary actions to get an initial feeling of what I was faced with. The first thing I tried was the command ldd to try to figure out whether any of the files contained in the image were dynamically linked to any binary file:

```
[root@LinuxForensics sample]# ldd ../image/*
../image/Acceptable_Encryption_Policy.doc:
        not a dynamic executable
../image/Information_Sensitivity_Policy.doc:
        not a dynamic executable
../image/Internal_Lab_Security_Policy1.doc:
        not a dynamic executable
../image/Internal_Lab_Security_Policy.doc:
        not a dynamic executable
../image/Password_Policy.doc:
        not a dynamic executable
../image/Remote_Access_Policy.doc:
        not a dynamic executable
```

But it is clear that .doc files are not executable files requiring any library.
Then the second thing I tried was the command file to check whether the files where again identified only as MS Office documents.

```
[root@LinuxForensics sample]# file ../image/*
../image/Acceptable_Encryption_Policy.doc:   Microsoft Office Document
../image/Information_Sensitivity_Policy.doc: Microsoft Office Document
../image/Internal_Lab_Security_Policy1.doc:  Microsoft Office Document
../image/Internal_Lab_Security_Policy.doc:   Microsoft Office Document
../image/Password_Policy.doc:                Microsoft Office Document
../image/Remote_Access_Policy.doc:           Microsoft Office Document
```

1.2.2 Strings analysis

I also felt the need, as the files where related then with text, to perform a strings command in the image.

```
[root@LinuxForensics sample]# strings ../sample/fl-260404-RJL1.img
>../sample/stringsimage
```

And at that moment is when I started to discover interesting things (and this is always encouraging for the forensic investigator). The file with the result of the strings command had 91KB and, among strings that had initially no sense, I already found this:

So, as a plain normal floppy disk, I will then have to use FAT12 as the appropriate option for autopsy to work fine.

```
mkdosfs
@RJL        FAT12
This is not a bootable disk.  Please insert a bootable floppy and
press any key to try again ...
"!""#B"%b"'
#1"#3B#5b#7
$A"$CB$Eb$G
%Q"%SB%Ub%W
```

In what it seemed as part of the file allocation table, I could see this data:

```
AMSHELLDLL
INFORM~1DOC
INTERN~1DOC
INTERN~2DOC
PASSWO~1DOC
REMOTE~1DOC
ACCEPT~1DOC
NDEX   HTM
```

Where, together with the six doc files already identified just by listing the content of the floppy, I also found two interesting names: a dynamically linked library (typical extension for elements of programs in the Windows world) with a name containing at least the letters ‘amshell’ and a possible index.htm file.

As I continued to read the strings output file, I also found more interesting traces linked with the previously mentioned dll:

```
ll\SheCamouflageShell
ShellExt
VB5!
CamShell
```

```
BitmapShellMenu
CamouflageShell
CamouflageShell
Shell_Declares
Shell_Functions
ShellExt
modShellRegistry
kernel32
lstrcpyA
lstrlenA
ole32.dll
CLSIDFromProgID
StringFromGUID2
ReleaseStgMedium
```

And a little bit afterwards, I could also read a string like this one:

```
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb
```

Apparently something named Camouflage was installed in a typical Windows computer. This program seems to be a piece of code in VB5.

Finally, together with the real text appearing in the doc files, I could also see lines like these:

```
2$2*20262<2B2H2N2T2Z2`2f2l2r2x2~2
3 3&3,32383>3D3J3P3V3\3b3h3n3t3z3
4"4(4.444:4@4F4L4R4Z4_4 54585P5X5l5p5x5
5@6T6X6`6p6
7 7(70787@7H7P7X7`7h7p7x7
8 8(80888D8H8T8X8\8h8x8
9 9$9(9,9<9@9D9H9L9P9p9t9x9|9
:0<<<@<L<h<x<
=$=,=4=T=X=\=`=
?8?<?D?Q?\?a?
0$0(000=0H0M0|0
```

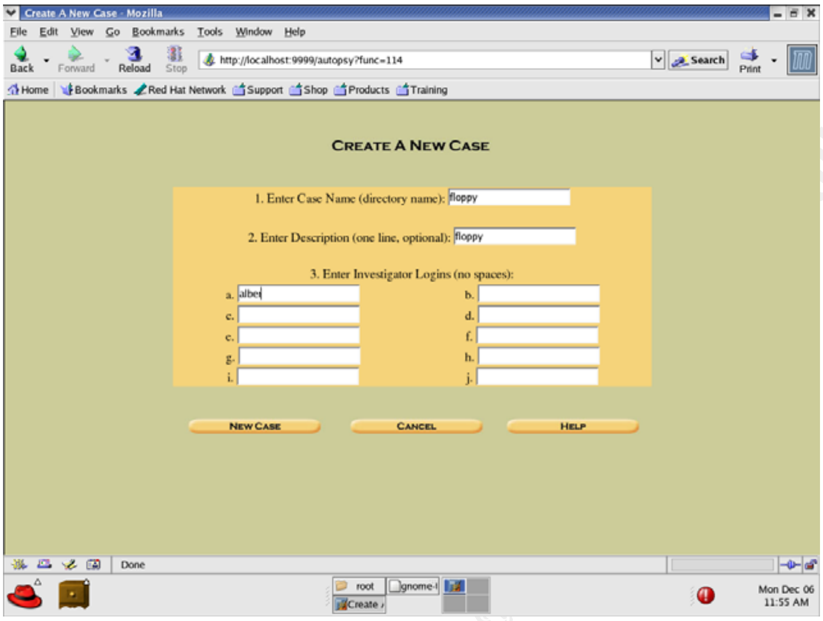
Although it is very soon to say that these lines belong to a cryptographic output, at least I can say that these uniform patterns are usually not found within doc files, at least not in doc files that do not have any kind of drawings or slides or graphs inserted.

1.3 *Image details using autopsy*

1.3.1 Creating the case

With these initially identified traces, now I start autopsy and I mount the floppy disk image into autopsy. In order to do this, it is important to remember that autopsy will work with the image file directly (for convenience, the image file has the extension .img.

I create myself a case and a host and I attach the floppy image to autopsy.


Create A New Case - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=114
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
root gnome-
Create
Mon Dec 06 11:55 AM

CREATE A NEW CASE

1. Enter Case Name (directory name): floppy

2. Enter Description (one line, optional): floppy

3. Enter Investigator Logins (no spaces):

a. alber
b.
c.
d.
e.
f.
g.
h.
i.
j.

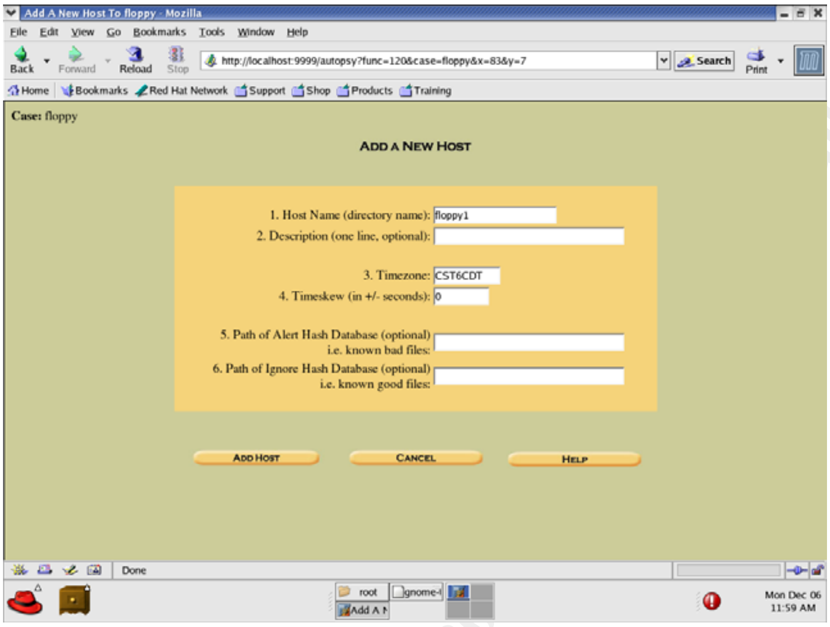
NEW CASE CANCEL HELP

This is the output I receive from autopsy:

Creating Case: floppy
Case directory (/forensics//floppy/) created
Configuration file (/forensics/floppy/case.aut) created
Investigators added
Case Details
Name: floppy
Description: floppy
Created: Mon Dec 6 11:56:16 2004
Directory: /forensics//floppy/
Images Dir: images
Output Dir: output
Report Dir: reports
Log Dir: logs
Investigators: alber
Linking /home/alber/foren1/fl-260404-RJL1.img to
/forensics//floppy/floppy1//images/fl-260404-RJL1.img

Calculating MD5 of images/fl-260404-RJL1.img (this could take a while)
Current MD5: D7641EB4DA871D980ADBE4D371EDA2AD
Image: /home/alber/foren1/fl-260404-RJL1.img added to config file as images/fl-260404-RJL1.img

I check that the md5 checksum provided by autopsy coincides with the md5 checksum I already performed on the file *fl-260404-RJL1.img* in the previous section of this assignment.
In autopsy the process to analyse an image is always the following: you create a case, then you add a host to that case:


Add A New Host To floppy - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
root gnome-t
Add A N
Mon Dec 06
11:59 AM

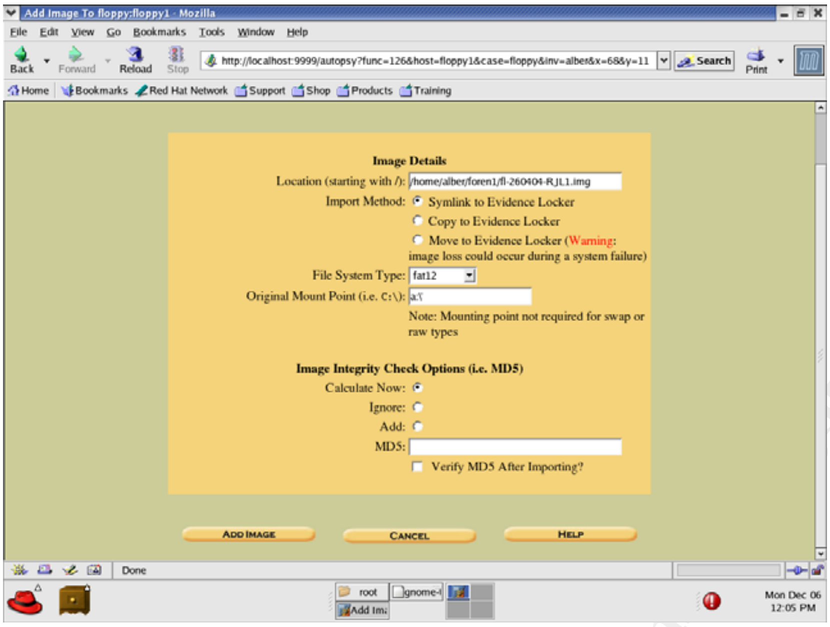
http://localhost:9999/autopsy?func=120&case=floppy&x=83&y=7

Case: floppy

ADD A NEW HOST

1. Host Name (directory name): floppy1
2. Description (one line, optional):
3. Timezone: CST6CDT
4. Timeskew (in +/- seconds): 0
5. Path of Alert Hash Database (optional) i.e. known bad files:
6. Path of Ignore Hash Database (optional) i.e. known good files:

And then you add images of the host to the case as it is shown in the next screen shot:



Add Image To floppy:floppy1 - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=126&host=floppy1&case=floppy&inv=alber&x=68&y=11
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training

Image Details

Location (starting with /): /home/alber/foren1/fl-260404-RJL1.img

Import Method: ● Symlink to Evidence Locker
○ Copy to Evidence Locker
○ Move to Evidence Locker (Warning: image loss could occur during a system failure)

File System Type: fat12

Original Mount Point (i.e. C:\): a:\

Note: Mounting point not required for swap or raw types

Image Integrity Check Options (i.e. MD5)

Calculate Now: ●
Ignore: ○
Add: ○
MD5:
□ Verify MD5 After Importing?

1.3.2 First idea about the different file types in the floppy

I executed sorter from autopsy to get comprehensive information about the types of files that were stored in the floppy image.

```
Executing: sorter -h -m 'a:\' -d '/forensics//floppy-disk-1/floppy1/output/sorter-fl-260404-RJL1.img/' -f fat16 '/forensics//floppy-disk-1/floppy1/images/fl-260404-RJL1.img'
Loading Allocated File Listing
Processing 8 Allocated Files and Directories
100%
Loading Unallocated File Listing
Processing 5 Unallocated meta-data structures
100%
All files have been saved to: /forensics//floppy-disk-1/floppy1/output/sorter-fl-260404-RJL1.img/
Output can be found by viewing: /forensics//floppy-disk-1/floppy1/output/sorter-fl-260404-RJL1.img//index.html
(Future versions of Autopsy will have built-in viewing capabilities)
Results Summary
Images
    * /forensics//floppy-disk-1/floppy1/images/fl-260404-RJL1.img
Files (13)
    * Allocated (8)
    * Unallocated (5)
Files Skipped (3)
    * Non-Files (3)
    * '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)
```

Why sorter says that there are 6 document files and 4 text files in the floppy image? Different answers can be given. I would dare say that from the 6 document files, 4 of them are text-based but something strange happen in the other two (more to come in the following sections).

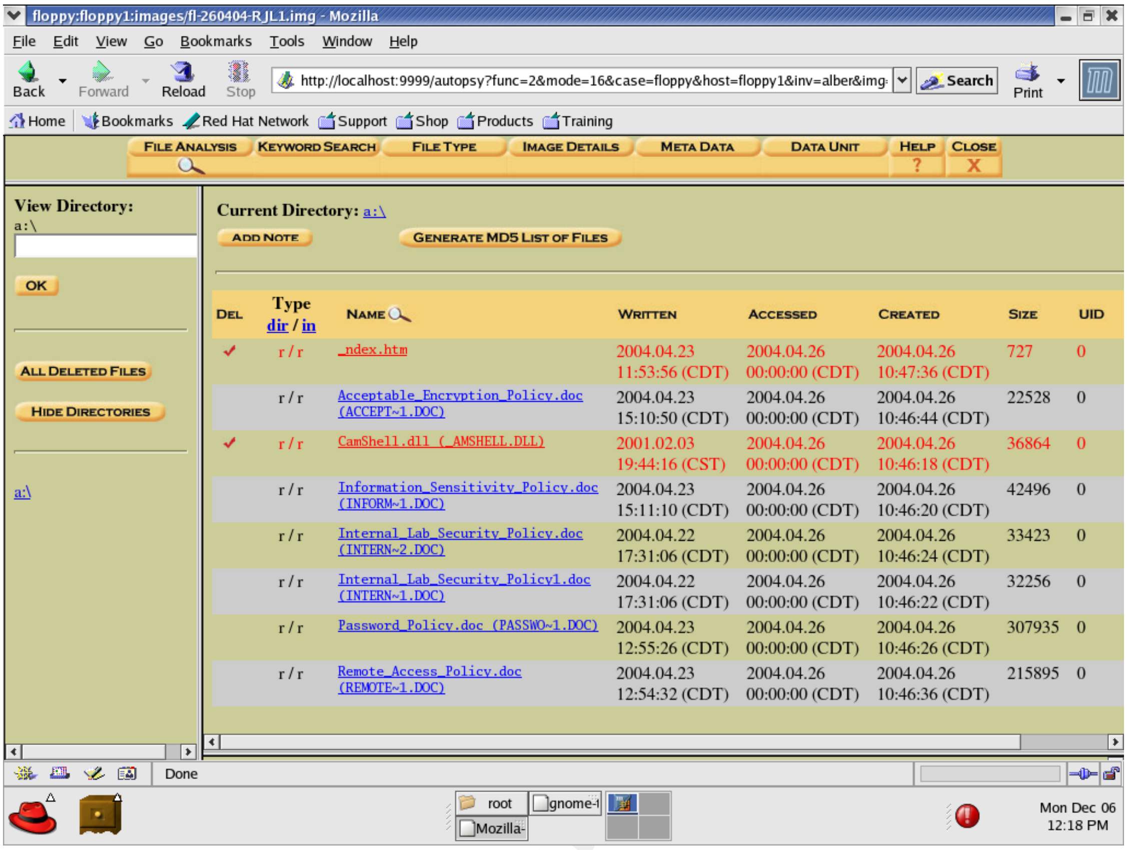
1.3.3 Listing of all the files in the image

Using the file analysis functionality in autopsy and requesting all files (including the deleted ones) to be showed, I reach the information present in the following screenshot. Together with the six doc files already mentioned before, there were two files in the floppy that were deleted: an index.htm file and a more interesting dll file named CamShell.dll with a size of 36 KB.

The list of all files in the image is the following:
Index.htm (deleted)
Camshell.dll (deleted)
Acceptable_Encryption_Policy.doc:
Information_Sensitivity_Policy.doc:
Internal_Lab_Security_Policy1.doc:
Internal_Lab_Security_Policy.doc:
Password_Policy.doc:
Remote_Access_Policy.doc:

1.3.4 Name of the program used by Mr.Leszczynski

The name of the file used by Mr. L (from now on, Mr. Leszczynski will be named Mr. L in this assignment) was Camshell.dll (as the reader will discover in the following section, I discovered that Camshell.dll is a dynamically linked library from the program Camouflage).

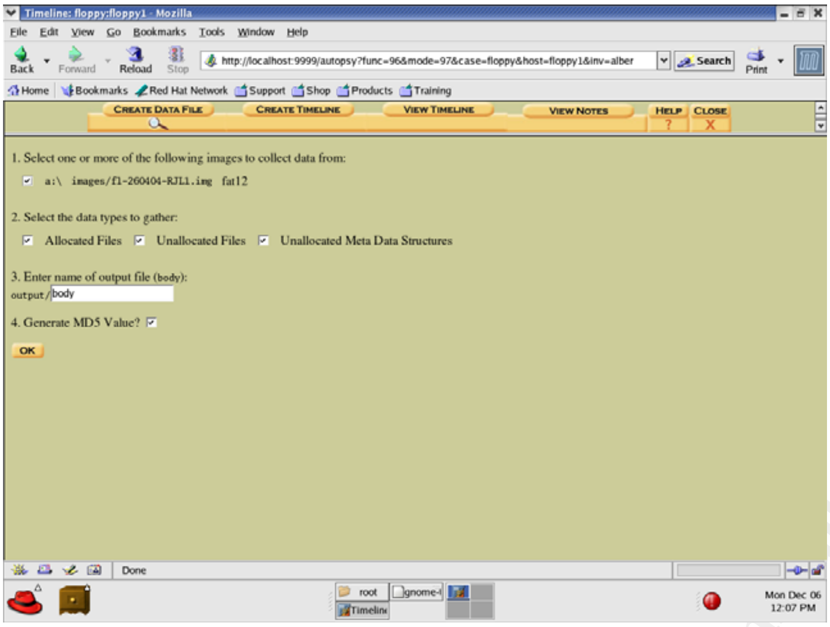

floppy:floppy1:images/fl-260404-RJL1.img - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=16&case=floppy&host=floppy1&inv=alber&img
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
View Directory:
a:\
OK
ALL DELETED FILES
HIDE DIRECTORIES
a:\
ADD NOTE
GENERATE MD5 LIST OF FILES
Done
root gnome-t Mozilla-
Mon Dec 06 12:18 PM

Current Directory: a:\

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID
✓	r / r	_ndex.htm	2004.04.23 11:53:56 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:47:36 (CDT)	727	0
	r / r	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)	2004.04.23 15:10:50 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:44 (CDT)	22528	0
✓	r / r	CamShell.dll (_AMSHELL.DLL)	2001.02.03 19:44:16 (CST)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:18 (CDT)	36864	0
	r / r	Information_Sensitivity_Policy.doc (INFORM~1.DOC)	2004.04.23 15:11:10 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:20 (CDT)	42496	0
	r / r	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)	2004.04.22 17:31:06 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:24 (CDT)	33423	0
	r / r	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)	2004.04.22 17:31:06 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:22 (CDT)	32256	0
	r / r	Password_Policy.doc (PASSWO~1.DOC)	2004.04.23 12:55:26 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:26 (CDT)	307935	0
	r / r	Remote_Access_Policy.doc (REMOTE~1.DOC)	2004.04.23 12:54:32 (CDT)	2004.04.26 00:00:00 (CDT)	2004.04.26 10:46:36 (CDT)	215895	0

1.3.5 Timeline and MAC information

Once I have properly installed the floppy image into autopsy, now I proceed to analyse it. I use autopsy GUI to produce a timeline of the image. This entails a three-step process. The first step consists of creating the data file including allocated, unallocated and unallocated meta data structures:



Timeline: floppy:floppy1 - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=96&mode=97&case=floppy&host=floppy1&inv=alber
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
CREATE DATA FILE
CREATE TIMELINE
VIEW TIMELINE
VIEW NOTES
HELP ?
CLOSE X

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

☑ a:\ images/fl-260404-RJL1.img fat12

2. Select the data types to gather:

☑ Allocated Files ☑ Unallocated Files ☑ Unallocated Meta Data Structures

3. Enter name of output file (body):
output/body

4. Generate MD5 Value? ☑

OK

Done
root
gnome-t
Timeline
Mon Dec 06
12:07 PM

file activity time lines
Running fls -r -m on images/fl-260404-RJL1.img
Running ils -m on images/fl-260404-RJL1.img
Body file saved to /forensics//floppy/floppy1/output/body
Entry added to host config file
Calculating MD5 Value
MD5 Value: 181BDD9CEC5B30CF7B27B6DE9A25DD58

The second step actually creates a file with the timeline:

Creating Timeline using all dates (Time Zone: CST6CDT)
Timeline saved to /forensics//floppy/floppy1/output/timeline
Entry added to host config file
Calculating MD5 Value
MD5 Value: 87DB1B75C3C5E1FD3A79D61A5328F84E

Having a look at the timeline, it can be concluded that on Friday April 23 2004 the six doc files were modified and on Monday April 26 2004 the six files seem to have been accessed at around 1:00 AM and again created at around 10:46 AM.

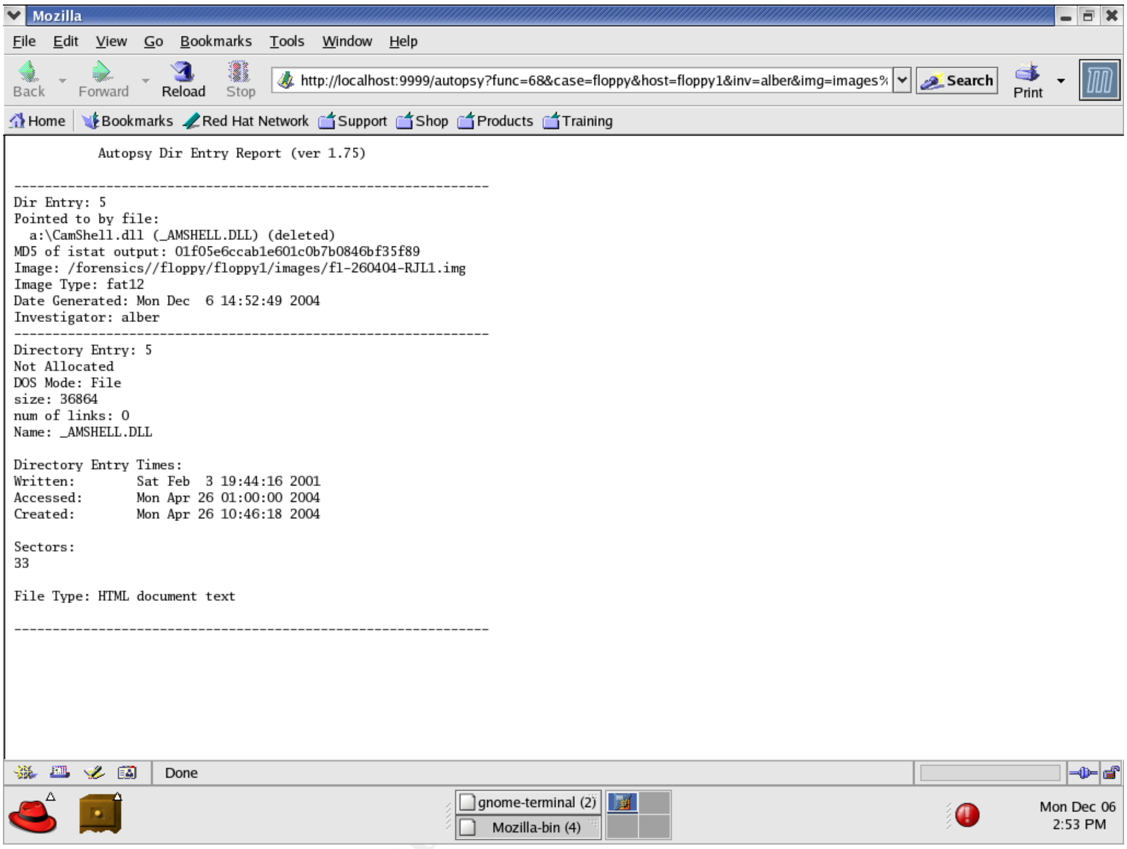
The timeline file has been attached to the annex of this assignment.

1.3.6 File owners

In a floppy disk formatted with a fat12 file system, file owners and user groups are concepts that do not apply. Using Unix terminology, all files are shown in autopsy as UID 0 and GID 0. They will be owned by the user that mounts the floppy disk.

1.3.7 File size

All file sizes in bytes can be seen in one of the screenshots inserted previously. According to autopsy, Camshell.dll has 36864 bytes. The following screenshot shows the autopsy dir entry report (version 1.75) for that file.


Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=6&&case=floppy&host=floppy1&inv=alber&img=images%
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
gnome-terminal (2)
Mozilla-bin (4)
Mon Dec 06
2:53 PM

```
          Autopsy Dir Entry Report (ver 1.75)

--------------------------------------------------------------
Dir Entry: 5
Pointed to by file:
  a:\CamShell.dll (_AMSHELL.DLL) (deleted)
MD5 of istat output: 01f05e6ccab1e601c0b7b0846bf35f89
Image: /forensics//floppy/floppy1/images/f1-260404-RJL1.img
Image Type: fat12
Date Generated: Mon Dec  6 14:52:49 2004
Investigator: alber
--------------------------------------------------------------
Directory Entry: 5
Not Allocated
DOS Mode: File
size: 36864
num of links: 0
Name: _AMSHELL.DLL

Directory Entry Times:
Written:        Sat Feb  3 19:44:16 2001
Accessed:       Mon Apr 26 01:00:00 2004
Created:        Mon Apr 26 10:46:18 2004

Sectors:
33

File Type: HTML document text

--------------------------------------------------------------
```

1.3.8 MD5 hash of the file

This interest of this section relies on the information mismatch between the md5 value of CamShell.dll provided by autopsy and the real md5 value of CamShell.dll. Both values do not coincide because the first 8 sectors of the file were occupied by another file that was also deleted afterwards.

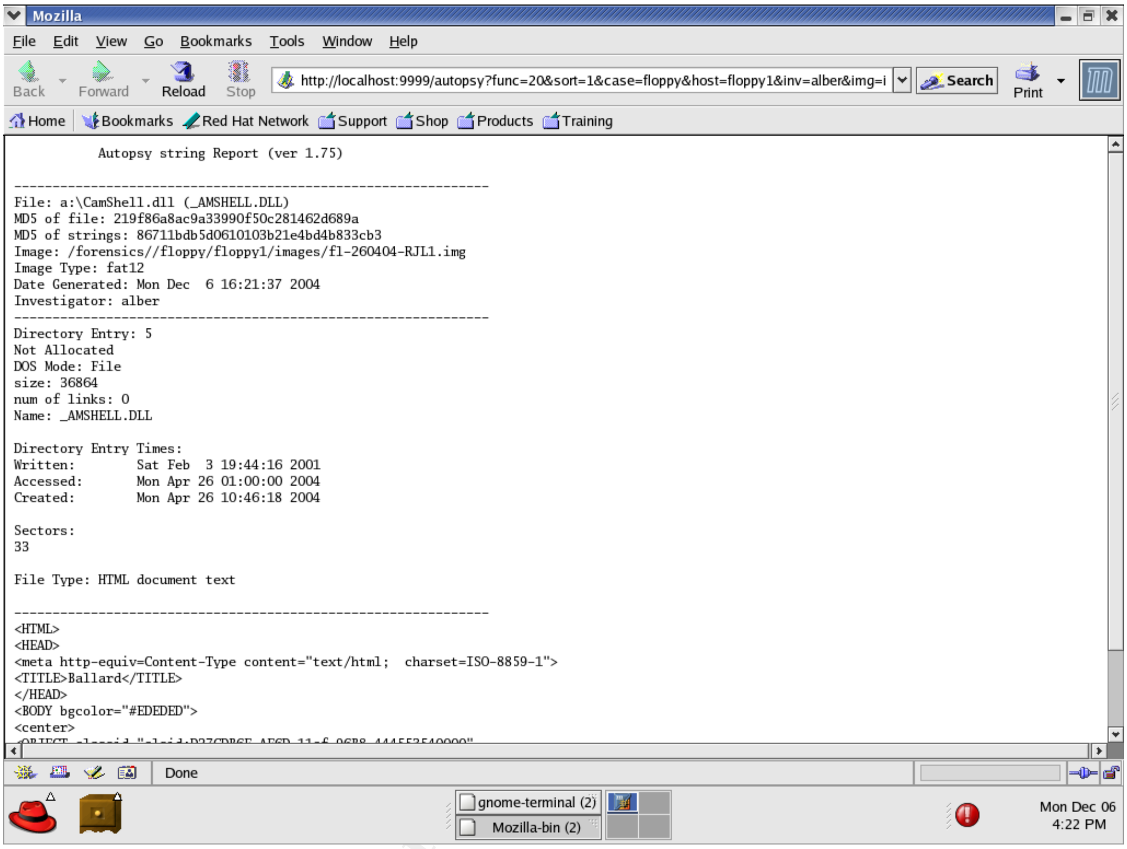
Index.htm actually occupied sectors 33 to 40, containing sectors 33 and 34 actual html data and sectors 35 to 40 only zeros. As CamShell.dll actually occupied 28 sectors starting from sector 33 (or at least that it was autopsy shows), I am afraid that the entire dll cannot be retrieved from the floppy disk image.

Therefore, although the size (36864 bytes) and creation date and time of the file (Saturday, February 03, 2001, 7:44:16 PM) coincide with the original

CamShell.dll that can be found on the Internet, it is not the case for the md5 value.

I will show both md5 checksum values mentioned in this section:

The first one is the md5 value showed by autopsy:



Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=20&sort=1&case=floppy&host=floppy1&inv=alber&img=i
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
Done
gnome-terminal (2)
Mozilla-bin (2)
Mon Dec 06
4:22 PM

```
            Autopsy string Report (ver 1.75)

---------------------------------------------------------------
File: a:\CamShell.dll (_AMSHELL.DLL)
MD5 of file: 219f86a8ac9a33990f50c281462d689a
MD5 of strings: 86711bdb5d0610103b21e4bd4b833cb3
Image: /forensics//floppy/floppy1/images/fl-260404-RJL1.img
Image Type: fat12
Date Generated: Mon Dec  6 16:21:37 2004
Investigator: alber
---------------------------------------------------------------
Directory Entry: 5
Not Allocated
DOS Mode: File
size: 36864
num of links: 0
Name: _AMSHELL.DLL

Directory Entry Times:
Written:        Sat Feb  3 19:44:16 2001
Accessed:       Mon Apr 26 01:00:00 2004
Created:        Mon Apr 26 10:46:18 2004

Sectors:
33

File Type: HTML document text

---------------------------------------------------------------
<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"
```

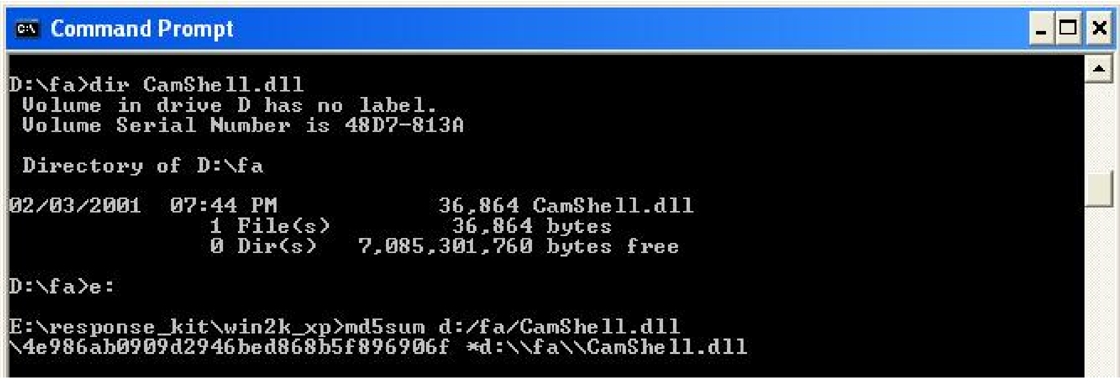
It is interesting to see (at the bottom of the screenshot), how the html text of the file index.htm can be found in CamShell.dll.

The second md5 I would like to show here is the md5 value of the real CamShell.dd. I also take the opportunity to show the creation date and time and the size of the file.

Again, the md5sum executable I use to produce the file checksum is the one located in the SANS Track 8 response kit cdrom.

An idea to check that it is really the same file is the following: we could take a subset of the sectors occupied by the file (any subset not containing the first sector would be appropriate) in both files, the one in the seized image and the one downloaded from Internet. Next, we could check that both subsets really produce the same checksum.

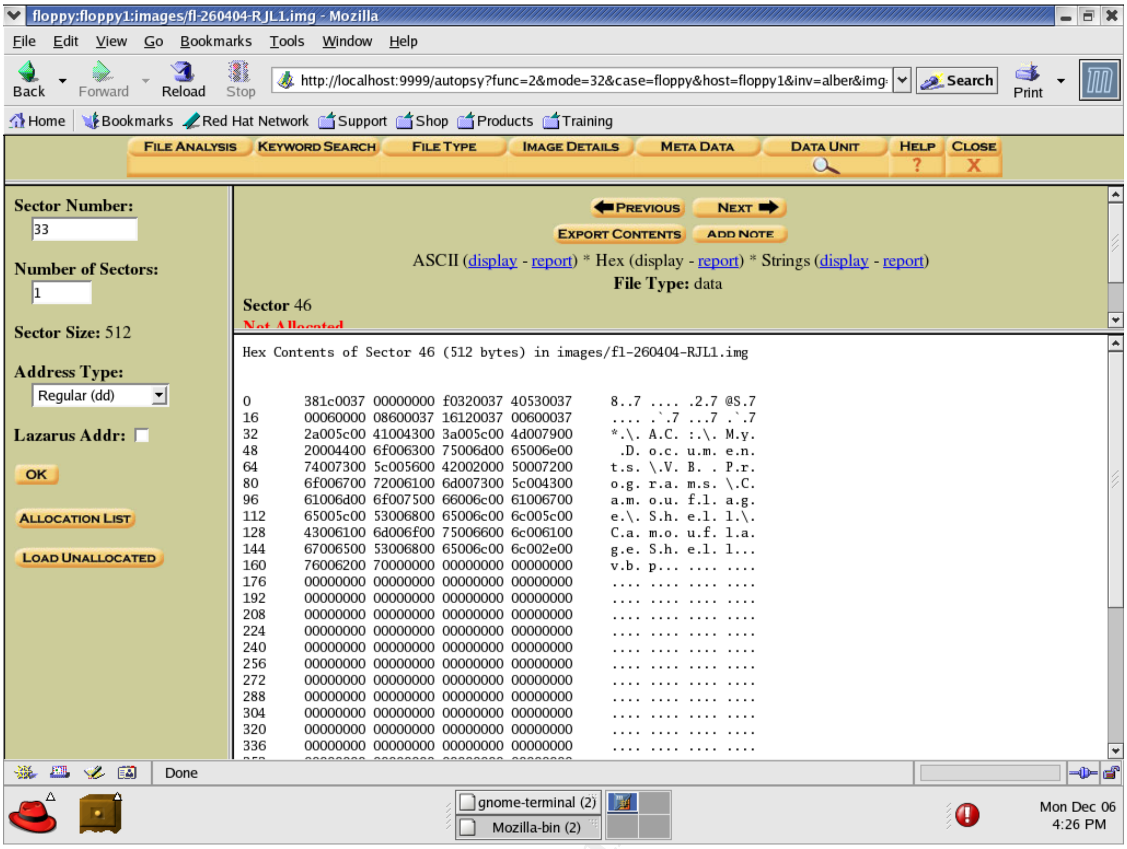
Anyway, as it will be demonstrated in the next sections using alternative ways, I decided not to further work on this idea.



```
D:\fa>dir CamShell.dll
 Volume in drive D has no label.
 Volume Serial Number is 48D7-813A

 Directory of D:\fa

02/03/2001  07:44 PM            36,864 CamShell.dll
               1 File(s)         36,864 bytes
               0 Dir(s)   7,085,301,760 bytes free

D:\fa>e:

E:\response_kit\win2k_xp>md5sum d:/fa/CamShell.dll
\4e986ab0909d2946bed868b5f896906f *d:\\fa\\CamShell.dll
```

1.3.9 Keywords found associated with the program

I link this section with one of the very first sections where I already performed a strings study of the full floppy image. There, interesting keywords were already identified (most of them certainly coming from the CamShell.dll file). Let's see whether they were really coming from the Camouflage dll.


floppy:floppy1:images/fl-260404-RJL1.img - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=32&case=floppy&host=floppy1&inv=alber&img
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
PREVIOUS NEXT
EXPORT CONTENTS ADD NOTE
OK
ALLOCATION LIST
LOAD UNALLOCATED
Done
gnome-terminal (2)
Mozilla-bin (2)
Mon Dec 06
4:26 PM

Sector Number: 33

Number of Sectors: 1

Sector Size: 512

Address Type: Regular (dd)

Lazarus Addr: □

ASCII (display - report) * Hex (display - report) * Strings (display - report)

File Type: data

Sector 46

Not Allocated

```
Hex Contents of Sector 46 (512 bytes) in images/fl-260404-RJL1.img


0       381c0037 00000000 f0320037 40530037    8..7 .... .2.7 @S.7
16      00060000 08600037 16120037 00600037    .... .`.7 ...7 .`.7
32      2a005c00 41004300 3a005c00 4d007900    *.\. A.C. :.\. M.y.
48      20004400 6f006300 75006d00 65006e00     .D. o.c. u.m. e.n.
64      74007300 5c005600 42002000 50007200    t.s. \.V. B. . P.r.
80      6f006700 72006100 6d007300 5c004300    o.g. r.a. m.s. \.C.
96      61006d00 6f007500 66006c00 61006700    a.m. o.u. f.l. a.g.
112     65005c00 53006800 65006c00 6c005c00    e.\. S.h. e.l. l.\.
128     43006100 6d006f00 75006600 6c006100    C.a. m.o. u.f. l.a.
144     67006500 53006800 65006c00 6c002e00    g.e. S.h. e.l. l...
160     76006200 70000000 00000000 00000000    v.b. p... .... ....
176     00000000 00000000 00000000 00000000    .... .... .... ....
192     00000000 00000000 00000000 00000000    .... .... .... ....
208     00000000 00000000 00000000 00000000    .... .... .... ....
224     00000000 00000000 00000000 00000000    .... .... .... ....
240     00000000 00000000 00000000 00000000    .... .... .... ....
256     00000000 00000000 00000000 00000000    .... .... .... ....
272     00000000 00000000 00000000 00000000    .... .... .... ....
288     00000000 00000000 00000000 00000000    .... .... .... ....
304     00000000 00000000 00000000 00000000    .... .... .... ....
320     00000000 00000000 00000000 00000000    .... .... .... ....
336     00000000 00000000 00000000 00000000    .... .... .... ....
```

The following is a list of keywords extracted from the dll:

ll\SheCamouflageShell
ShellExt
VB5!
CamShell
BitmapShellMenu
CamouflageShell
CamouflageShell
Shell_Declares
Shell_Functions
ShellExt
modShellRegistry
kernel32
lstrcpyA
lstrlenA

As I will explain in coming sections, I was unable to locate the source code in the Internet. To minimise this drawback, I will then appeal again to the strings study to try to understand and explain the functioning of this program.

1.4 Forensic details

1.4.1 Name and type of program used by Mr.L

This was the fun searching part of the assignment. The program Mr. L was using is Camouflage (more specifically, version 1.2.1). Camouflage is a simple steganography program that hides any file within any other file (preferably the destination file should not be a text file).

The last time it was used, according to the timeline obtained from autopsy (see Annex) was on Monday 26 April 2004:

```
Mon  Apr 26 2004 10:46:18 36864 ..c -/-rwxrwxrwx 0 0 5 a:\/CamShell.dll
(_AMSHELL.DLL) (deleted)
   36864 ..c -rwxrwxrwx 0 0 5 <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5 >
```

1.4.2 Investigation line

How did I find the program linked to CamShell.dll? I first performed a simple query in google (http://www.google.com) with the name CamShell.dll. The results were the first piece in this puzzle. I was directed to a trance-music discussion forum (http://www2.tranceaddict.com/forums/archive/topic/79627-1.html) where a user named ‘raver31’ posted a question referring to CamShell.dll.

In the same conversation, a user called ‘frystyler’ says that “…is a program that lets u hide files in jpg images…”. And finally, ‘tranceman 78’ mentions the name of Camouflage.

The following step was performing again a search in Google but this time with the name of the program, Camouflage. Here I had to restrict somehow the results since only with the word Camouflage I could not really find the right software-related web page. Therefore, I decided to use ‘download Camouflage’ in the google search and I found out http://camouflage.unifiction.com/Overview.html

There I could find a FAQ section (http://camouflage.unifiction.com/FAQ.html) and I could even download the software package.

1.4.3 Knowing Camouflage

I thought that SANS web site could very well be a place to search additional information about Camouflage (by that time, I only suspected what Mr.L could have done but I could not proof it yet).

I searched the word Camouflage in the search functionality in http://www.sans.org and voila! Two different papers appeared in the result list:

- A GSEC paper from 2001: Steganography: Past, present and future by James C. Judge.
URL: http://www.sans.org/rr/whitepapers/stenganography/552.php

- A GSEC paper from 2002: The ease of steganography and Camouflage by John Bartlett (17 March 2002).
URL: http://www.sans.org/rr/whitepapers/vpns/762.php

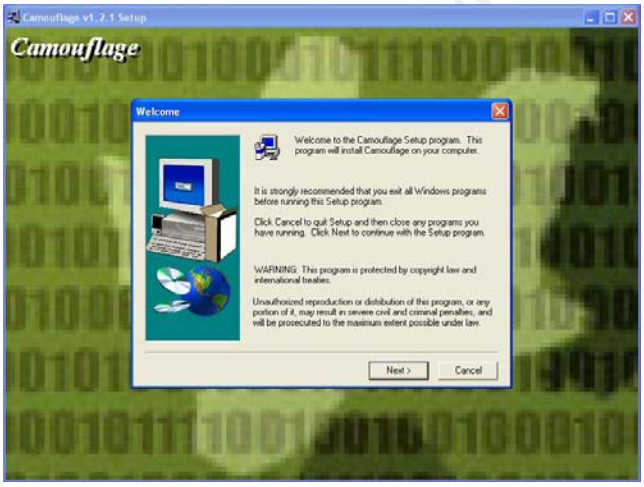
The first paper only mentions Camouflage as a possible tool for file steganography. In this paper is where I first find the original Camouflage web site: http://www.camouflagesoftware.com Unfortunately, currently this web site seems to be completely unavailable.

The second paper includes a fully-fledged description of the Camouflage software (luckily even the same version, or one compatible, as the one used by Mr.L). Bartlett's paper states two major points about Camouflage:

- Its user friendliness: in less than eight mouse clicks anyone can hide a file into another file.
- Its noisiness, plenty of traces testifying that Camouflage has been used in a determined box (e.g. numerous registry entries) and the size aspect, a telling feature that really shows clearly whether a file contains something hidden or not.

1.4.4 Step-by-step installation and analysis

In order to get to know Camouflage, I prepared a test-dummy XP box and I installed it there. Some screenshots of this process follow:



Welcome to the Camouflage Setup program. This program will install Camouflage on your computer.

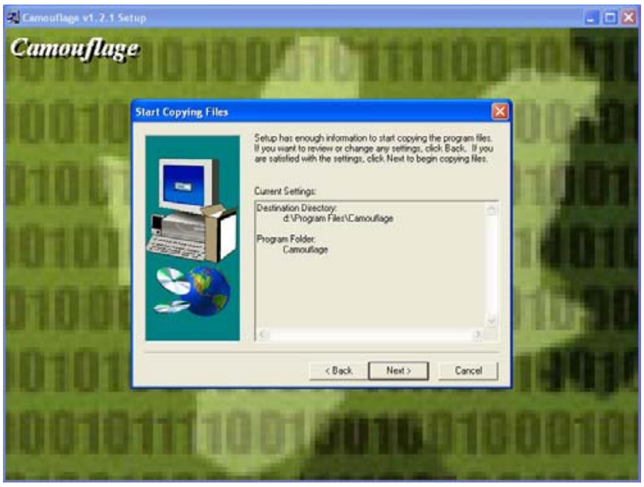
It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

The installation wizard is the typical installation process in Windows.



Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

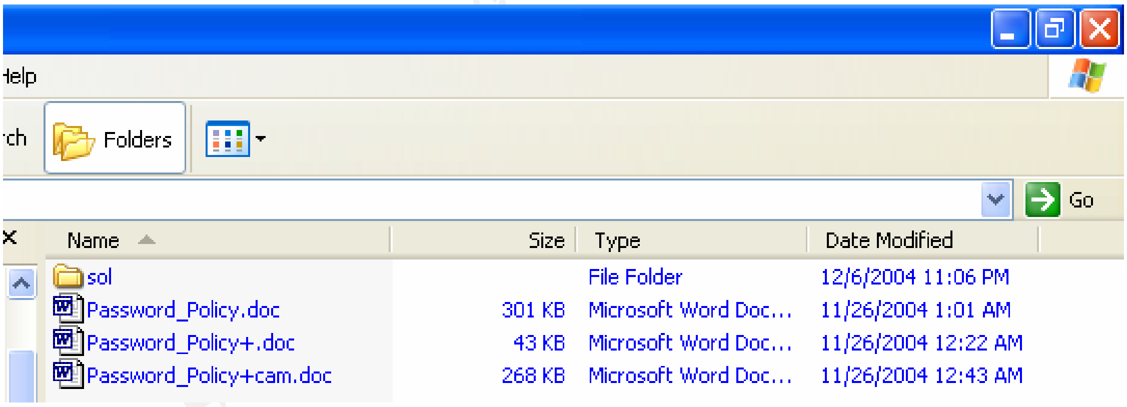
Destination Directory:
d:\Program Files\Camouflage

Program Folder:
Camouflage

Once the program is installed, I practice with it and I see how easy to use it is. If I now right-click in any file showed by the file explorer, I have two new context possibilities, camouflage or uncamouflage a file. While camouflaging a file, I can also choose a password that will be required when ‘uncamouflaging’ the file.

If I now have a look at the doc files stored in the floppy image and I play with their sizes, it is easy to identify that two of them have ‘something special’ in their inside. Additionally, if I open all of them with a tool different to Word, e.g. Notepad, I also see that two of them have an incredible amount of ‘scrumbled or encrypted’ characters at their end.

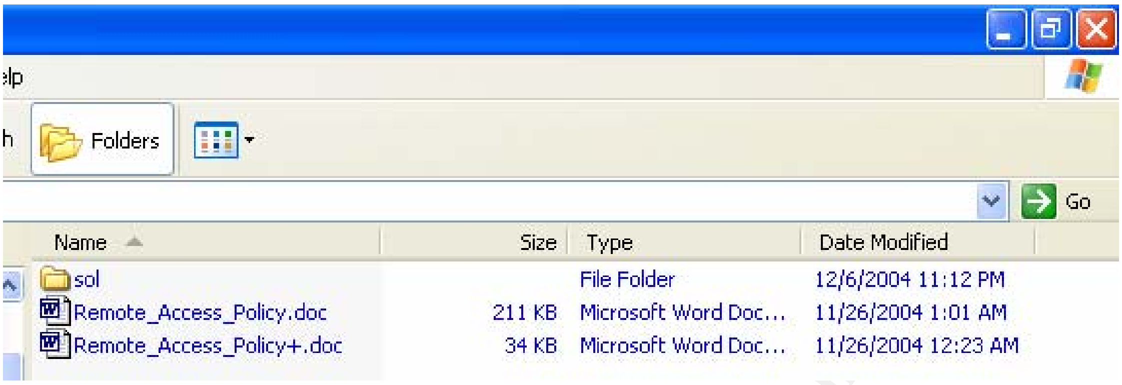
In the following file list, it is easy to detect interesting things:



Name	Size	Type	Date Modified
sol		File Folder	12/6/2004 11:06 PM
Password_Policy.doc	301 KB	Microsoft Word Doc...	11/26/2004 1:01 AM
Password_Policy+.doc	43 KB	Microsoft Word Doc...	11/26/2004 12:22 AM
Password_Policy+cam.doc	268 KB	Microsoft Word Doc...	11/26/2004 12:43 AM

Password_Policy is the original file stored in the floppy image. If I open it with Word and I only add a simple character in the file and afterwards I save the new file with the name Password_Policy+.doc, surprisingly the size of the new file decreases from 301 KB in the original Password_Policy file to some mere 43 KB.

The same happens with the file Remote_Access_Policy.doc. If I open it with Word and I add a simple character, then its sizes decreases from 211 KB to 34 KB.



Name	Size	Type	Date Modified
sol		File Folder	12/6/2004 11:12 PM
Remote_Access_Policy.doc	211 KB	Microsoft Word Doc...	11/26/2004 1:01 AM
Remote_Access_Policy+.doc	34 KB	Microsoft Word Doc...	11/26/2004 12:23 AM

So far, at least it can be stated that both files have something strange in their content. The next step I tried was to 'uncamouflage' these files, but interestingly enough, the software asked me for a password. A password that I did not know.

Camouflage adds blocks of data to be hidden just after the sequence in the file announcing the end of the doc file. For this reason, when a camouflaged file is opened by the application meant to be related to the 'recipient file' (in our case, the doc file), then all hidden information is deleted because the end of file code is rightly inserted by the respective application right after the end of the visible file content.

1.4.5 Breaking Camouflage

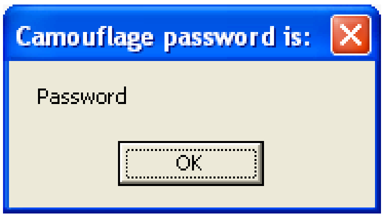
The next search I performed via google was 'breaking Camouflage' and voila it turned out that http://guillermito2.net/stegano/camouflage explained how Camouflage worked and demonstrated the weakness of its scrambling mechanism: in only eight pages he clearly shows that the 'scrumbled' output scheme produced by Camouflage does not depend on the password!

Even more, he identifies where the password is stored (always with an offset of -275 in decimal relative to the end of the file) and that an XOR operation between the password buffer and a key also stored in the file results in the password used to scramble the file.

I even downloaded a small utility from http://guillermito2.net/stegano/camouflage claiming to be able to recover password from camouflaged files (versions 1.1.1 and 1.2.1). I installed in my test-dummy XP workstation and I discovered what Mr. L got out of Ballard Industries.

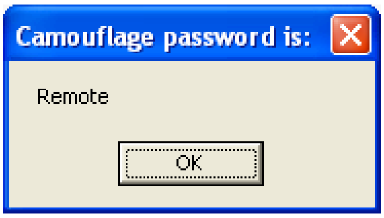
I only had to indicate to this utility the file I would like the password from and:

- This was the result for Password_Policy.doc:


Camouflage password is:

Password

- This was the result for Remote_Access_Policy.doc

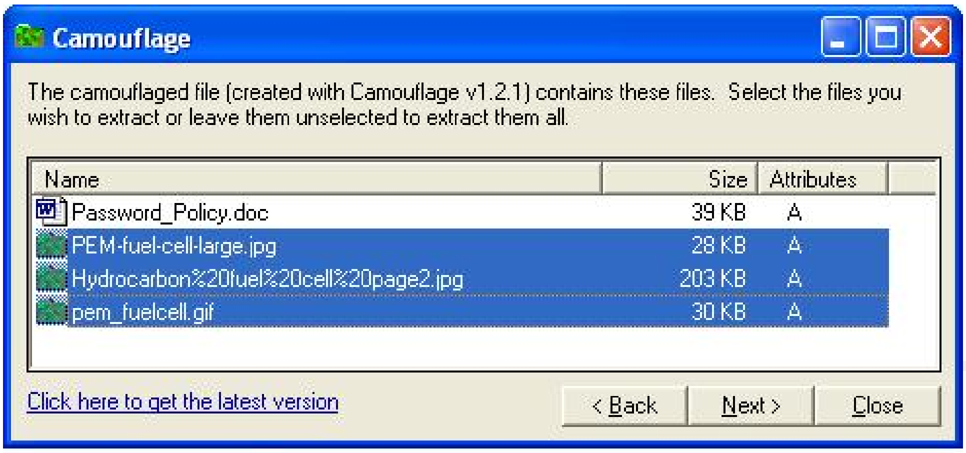

Camouflage password is:

Remote

1.4.6 Hidden information in Mr.L's floppy disk

Once I know the probable passwords, I try them in those files using my Camouflage XP installation. Password_Policy.doc has three hidden files:
- PEM-fuel-cell-large.jpg
- pem_fuelcell.gif
- Hydrocarbon%20fuel%20cell%20page2.jpg

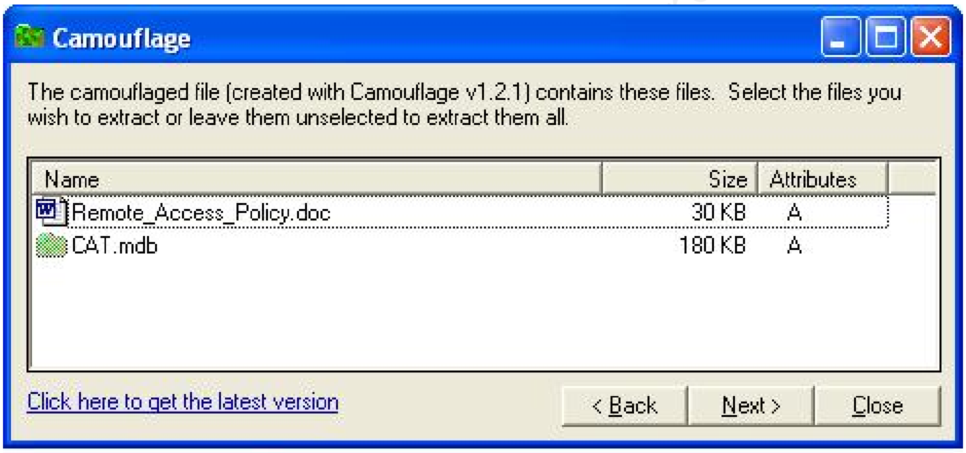
Remote_Access_Policy.doc had also one hidden file:
- CAT.mdb: An Access database with client information including name, phone number, company address, account and password (probably from a customer relationship management software used at Ballard).



The camouflaged file (created with Camouflage v1.2.1) contains these files. Select the files you wish to extract or leave them unselected to extract them all.

Name	Size	Attributes
Password_Policy.doc	39 KB	A
PEM-fuel-cell-large.jpg	28 KB	A
Hydrocarbon%20fuel%20cell%20page2.jpg	203 KB	A
pem_fuelcell.gif	30 KB	A

It is clear now that Mr.L was hiding cell-related information and customer data in the floppy disk. No surprise then that Rift Inc. could have built the same fuel battery which was once unique to Ballard.

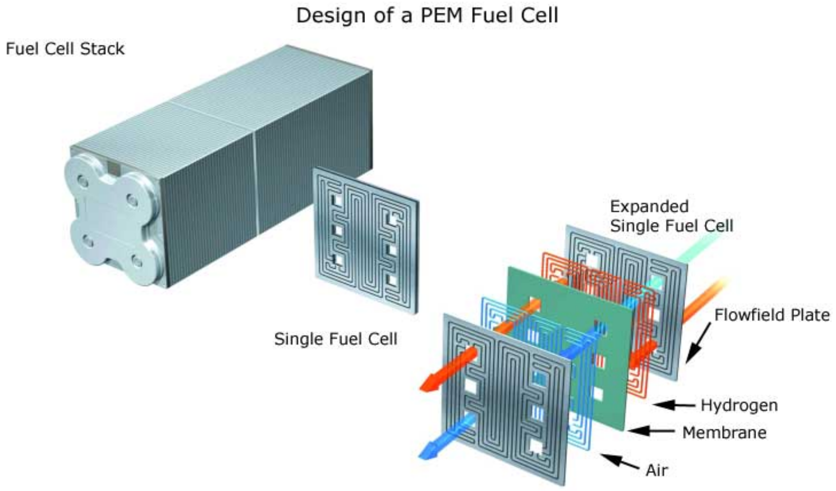


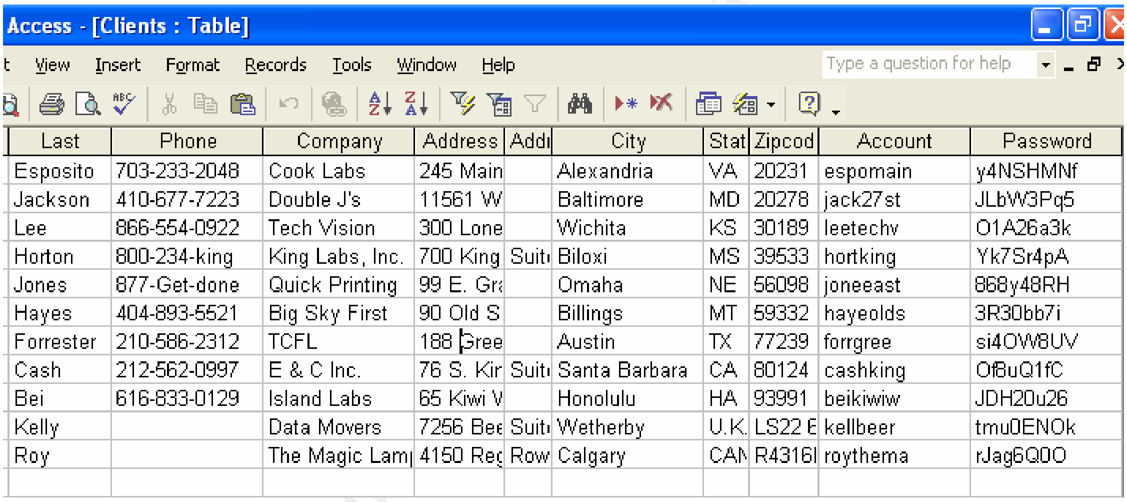
The camouflaged file (created with Camouflage v1.2.1) contains these files. Select the files you wish to extract or leave them unselected to extract them all.

Name	Size	Attributes
Remote_Access_Policy.doc	30 KB	A
CAT.mdb	180 KB	A

Here I also attached two screen shots from the hidden files. Screenshots from the other two files can be found in the annex:

- PEM-fuel-cell-large.jpg
- CAT.mdb




Access - [Clients : Table]
t View Insert Format Records Tools Window Help
Type a question for help

Last	Phone	Company	Address	Addı	City	Stat	Zipcod	Account	Password
Esposito	703-233-2048	Cook Labs	245 Main		Alexandria	VA	20231	espomain	y4NSHMNf
Jackson	410-677-7223	Double J's	11561 W		Baltimore	MD	20278	jack27st	JLbW3Pq5
Lee	866-554-0922	Tech Vision	300 Lone		Wichita	KS	30189	leetechv	O1A26a3k
Horton	800-234-king	King Labs, Inc.	700 King	Suit	Biloxi	MS	39533	hortking	Yk7Sr4pA
Jones	877-Get-done	Quick Printing	99 E. Gr		Omaha	NE	56098	joneeast	868y48RH
Hayes	404-893-5521	Big Sky First	90 Old S		Billings	MT	59332	hayeolds	3R30bb7i
Forrester	210-586-2312	TCFL	188 Gree		Austin	TX	77239	forrgree	si4OW8UV
Cash	212-562-0997	E & C Inc.	76 S. Kir	Suit	Santa Barbara	CA	80124	cashking	Of8uQ1fC
Bei	616-833-0129	Island Labs	65 Kiwi V		Honolulu	HA	93991	beikiwiw	JDH20u26
Kelly		Data Movers	7256 Bee	Suit	Wetherby	U.K.	LS22 E	kellbeer	tmu0ENOk
Roy		The Magic Lam	4150 Re	Row	Calgary	CAN	R4316I	roythema	rJag6Q0O

1.5 Program identification

As previously stated, the original Camouflage web site http://www.camouflagesoftware.com is not available anymore. I could not find the program's source code on the Internet. Alternatively, a demonstration that the program used by Mr. L was exactly Camouflage v1.2.1 is the reverse 'unhiding' process that I have described in the preceding sections.

Additionally, the strings study performed in the floppy image can very well tell what and how Camouflage works, as already described in the step by step analysis. A copy of the strings output has been enclosed in the annex of this assignment.

It is a Visual Basic piece of code using system functions input/output functions such as stringcopy, stringlength and get and select object. All these actions are required to get the file-to-hide as an input and attach it in the recipient-file making first a XOR operation with the chosen password, as described in http://guillermito2.net/stegano/camouflage.

In the previous link it can be clearly the usefulness of a very basic but powerful tool to do file forensics: an hexadecimal editor. Most of the conclusions reached in those pages are based on the hexadecimal dump of camouflaged and non-camouflaged files.

The philosophy behind the analysis is similar to the one applied in cryptoanalysis with known input: trying to find out the scrumbling process by making slight modifications in the input of the scrumbling tool and comparing the different outputs.

The code is using msvbvm60.dll (see strings study in the annex), this dll implements the Visual Basic Virtual Machine environment (this demonstrates that the code has originally been written in Visual Basic). I can also read in the strings dump that initially the program was installed in C:\My Documents\VB Programs\Camouflage\Shell\. This makes me recommend Ballard Industries Inc. to seize Mr. L's desktop, since it could very well be that the original software is still installed in his office PC. This way, an additional evidence supporting Mr. L's actions could be used by Ballard's legal services.

As a side learning point, I would like to stress here the recommendation for companies not to just simply allow their staff install any piece of software in their desktops. An agreed policy stating that and some technical measures contributing to the enforcement of this measure (such as, e.g. no admin rights given in desktops to employees) will be desirable.

1.6 Legal implications

The security guard at Ballard seized a floppy disk located in Mr. L's briefcase. It is proofed that the floppy disk contained hidden information property of Ballard such as a technical note on the hydrocarbon cell, two explanatory drawings of the proton exchange cell and a database with information from 11 customers including their username and password (probably to access a customer relationship management software).

Luckily, the security guard seized the floppy disk just when Mr. L was trying to take it out of Ballard's facilities. So, this time the proprietary information did not leave the building.

It is also demonstrated that the program used to hide the files was Camouflage. A dll named CamShell.dll was stored (and afterwards deleted) in the floppy disk.

The reverse 'unhiding' process was performed and the information was easily retrieved from the camouflaged files.

In order to proof that Camouflage was executed in a system owned by Ballard, it would be required checking whether e.g. Camouflage was installed in Mr. L's desktop.

It is not clear for me yet why the Camshell.dll was residing in the floppy disk. In order to install Camouflage, more files are required and there is also a more general question: why in the floppy disk and not in the desktop's hard disk? Probably because security at Ballard already prevent the installation of uncontrolled software by the users in their workstations.

I consider that Ballard is a US-based company. Therefore, US Law applies to this case. The first thing to check would be whether Ballard's computers could be considered "protected computers" by the Computer Fraud and Abuse Act (Title 18 of the United States Code, section 1030).

This would be the case if Ballard cells are sold outside the US. This way, it could be argued that Ballard's systems affect the foreign commerce of the United States. I will assume that this is the case for Ballard.

The interesting thing is that Mr. L's actions cannot be considered as "damage" according to the Act because they did not affect the integrity of availability of Ballard's information.
Anyway, it was an attack purely against the confidentiality of the information, and the Federal Computer Fraud & Abuse Act prohibits intentional access to a protected computer in excess of authorization to obtain information aimed to get either commercial advantage or private financial gain or if the information is worth more than $5000. I think this scenario applies to our case.

It the Court considers that Ballard's proprietary information is part of a "protected computer", Mr. L could face a sentence varying from a fine and one year imprisonment (if he is a first-time offender) to a fine and 10 years (if it is not the first similar offense and it has been committed with certain listed motives).

If Ballard decides not to formally sue Mr. L for any reason (e.g. lack of time and additional resources required to conduct the suit)., at least I consider that Ballard made Mr. L sign when he joined the company a confidentiality statement in line with Ballard's internal information security policy. This internal policy should cover the prohibition to take out from the company any proprietary information (and especially to provide that information to any competitor).

Would that be the case, the Ballard could fire Mr. L stating that it was proof that he did not respect the policy and he tried to take proprietary and confidential information out of the company using a steganography program to avoid being caught.

1.7 Additional information

As already described throughout the preceding sections, I used more than three outside sources of information for my research. I proceed to enumerate them in this section:
Firstly the most important information gathering tool (or hacking tool) nowadays:
- http://www.google.com

Secondly, the very first place I found where CamShell.dll was mentioned:
- http://www2.tranceaddict.com/forums/archive/topic/79627-1.html

Thirdly, web pages presenting Camouflage the tool:
- http://camouflage.unifiction.com/Overview.html
- http://camouflage.unifiction.com/FAQ.html

Fourthly, two SANS GSEC papers providing me nice tips:
- http://www.sans.org/rr/whitepapers/stenganography/552.php
- http://www.sans.org/rr/whitepapers/vpns/762.php

Finally, the Camouflage cracking study made by ‘guillermito’ in 2002, a paper really worth reading it.
- http://guillermito2.net/stegano/camouflage

And the small utility he also provides to retrieve passwords
http://guillermito2.net/stegano/camouflage/Camouflage_Password_Finder_02.zip

2. Part 2- Option 1: Perform forensic analysis on a system

This second part of the assignment aims to provide a step-by-step guide for forensic analysts when they are asked to investigate an unknown system and provide a jury with their assessment and final conclusions.

2.1 Synopsis of case facts

In order to find a system in an unknown state that I could use for this part of the assignment, I decided to ask a very close friend of mine for her laptop. I promised her I would give the laptop safe and sound back as soon as I finalized my forensic investigation and I also agreed with her that I would eliminate (or anonimize) any personal data I could find in her laptop.

At the same time, as a way for me to give something valuable back, I also committed myself to review the overall security state of her laptop and report to her any possible improvements that could be conducted in the laptop.

As a final initial remark, I will mention that my friend is not a computer-savvy. She uses her laptop mostly to browse through the Internet, write reports, elaborate spreadsheets, prepare leisure trips, write emails and eventually chat with friends. This was the main reason why I thought her laptop could very well represent an average Internet young user's box.

2.2 System description

My friend has always been interested in design and high-and products. Therefore, her laptop is an Apple 'iBook G4' with the following characteristics:

- M9388 model bought in December 2003 acquired in an Apple store.
- Processor: A PowerPC G4 working at 933 MHz.
- Physical memory extended to 640 MB.
- Hard disk Ultra ATA/100 with 60 GB.
- DVD-ROM/CD-RW slot-combo drive.
- XGA TFT 14.1", 1024x768 pixels.
- Graphic card: ATI Mobility Radeon 9200 card with 32 MB DDR SDRAM.
- Ports: 1 Firewire 400 port, 2 USB 2.0, a VGA, S-video output.
- Stereo headphones output.
- Network card: Embedded Ethernet 10/100 Base T and a V.92 56K modem.
- A wireless Airport Extreme and Bluetooth module also embedded.
- Battery duration: Up to 6 hours.
- Dimensions: 32.3 x 25.9 x 3.4 cm.
- Weight: 2.7 kg.
- Initial software included: Mac OS X, Mail, iChat AV, Safari, Sherlock, Agenda, Quicktime, iLife (including iTunes, iPhoto and iMovie), iSync, iCal, DVD player, Appleworks, Mac OS Chess and Classic environment.

- Additionally, a 32 MB USB memory stick that she uses as a useful way to carry data between her personal laptop and her office desktop.

As mentioned before, this laptop is mostly used to browse through the Internet, write reports, elaborate spreadsheets, prepare leisure trips, write emails and eventually chat with friends.

2.3 Hardware

As this Apple iBook is a laptop, all its components such as battery, hard disk, communication modules and memory cards are embedded in the laptop case.
It would be possible to open the case and write the serial numbers of the different components down but I decided not to take that risk since I had to give the laptop back to its owner in perfect conditions.

Interestingly enough, no serial number appears externally in the laptop case. The only information I could gather was the following:

Model number A1055, for home and office use, copyright Apple 2003.
Additionally, the list of the components forming this laptop has already been attached in the previous section.

As a preventive measure, I took digital photos of the seized laptop and I kept it safe at my place during the whole investigation process, so that no one could access it.

2.4 Image media

In forensic terms, a disk image is a bit-for-bit copy of a drive (in contrast to a backup, which is a copy only of allocated space). In order to obtain a forensic image media of the hard drive of the Mac, I first follow the following step-by-step process:

- I ask my friend to log in with her usual username and password.
- I check in the Mac using Finder the size of the Macintosh HD (Capacity: 55.88 GB, 36.66 GB available, 19.21 GB used in the disk).
- Due to time constraints and available space in the Linux forensic workstation, it is not possible to obtain an image media of the complete hard disk, therefore I decide to first try to obtain 1048576 (1 M) blocks of 512 bytes using the switch –count in the dd command.
- I connect via a crossover Ethernet cable my Linux forensic workstation (to which I assigned the private IP address 192.168.2.1 subnet mask 255.255.255.0) and the Mac (to which I assign the private IP address 192.168.2.2 subnet mask 255.255.255.0).

- I check whether I already have IP connectivity between both boxes:

```
[root@LinuxForensics root]# ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=64 time=0.221 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=64 time=0.210 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=64 time=0.214 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=64 time=0.216 ms
64 bytes from 192.168.2.2: icmp_seq=5 ttl=64 time=0.218 ms
64 bytes from 192.168.2.2: icmp_seq=6 ttl=64 time=0.221 ms
--- 192.168.2.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 0.210/0.216/0.221/0.017 ms
```

And I also do the same thing from the Mac to the Linux box.

```
Dairy:~ suis$ ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1): 56 data bytes
64 bytes from 192.168.2.1: icmp_seq=0 ttl=64 time=0.355 ms
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.351 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.337 ms
^C
--- 192.168.2.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.337/0.347/0.355 ms
```

- I start a netcat listener in my forensic workstation:

```
[root@LinuxForensics linux_x86_static]# nc -l -p 31330 > /tmp/forensics/mac_hdisk.dd
```

- I try to launch the dd command in the Mac, I first try with a count of 1000. I learnt that the hard disk in a Mac system is usually /dev/rdisk0 in http://www.aplawrence.com/Bofcusm/2283.html . Normally /dev/ is created when the OS is installed.

```
Dairy:/dev alber$ dd if=/dev/rdisk0 count=1000 of=/tmp2/prueba8.dd
dd: /dev/rdisk0: Permission denied
```

But I need to be root to dd rdisk0:

```
Dairy:/dev alber$ su root
Password:
Dairy:/dev root# dd if=/dev/rdisk0 count=1000 of=/tmp2/test8.dd
1000+0 records in
1000+0 records out
512000 bytes transferred in 0.155505 secs (3292500 bytes/sec)
```

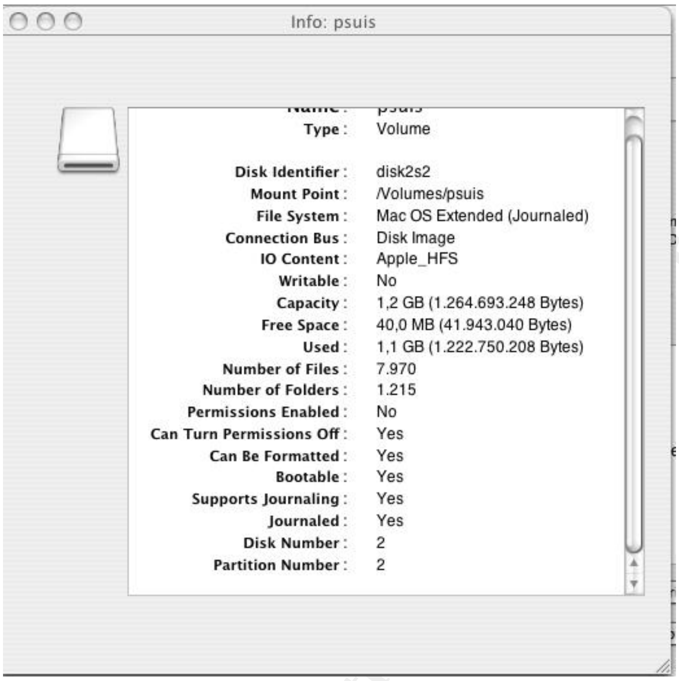
The message I get from the dd software coming in the Mac OS X (version Panther) is positive, the transfer can be done. I then try the following:

```
Dairy:/suis$ dd if=/dev/rdisk0 count=1048567 | nc 192.168.2.1 31330
```

As a comment, I would like to add that the same operation does not work if the link *Macintosh HD* and not */dev/rdisk0* is used. This is the message I got:

```
dd: /Volumes/Macintosh HD/Users: Operation not permitted
0+0 records in
0+0 records out
0 bytes transferred in 0.032288 secs (0 bytes/sec)
```

I adopt a pragmatic approach and then I use the Mac Disk Utility to create an image first within the Mac itself and, as Disk Utility only offers the possibility to create an image of a folder (in contrast to the switch ‘count’ in dd), I decide to create the image of my friend’s user folder.

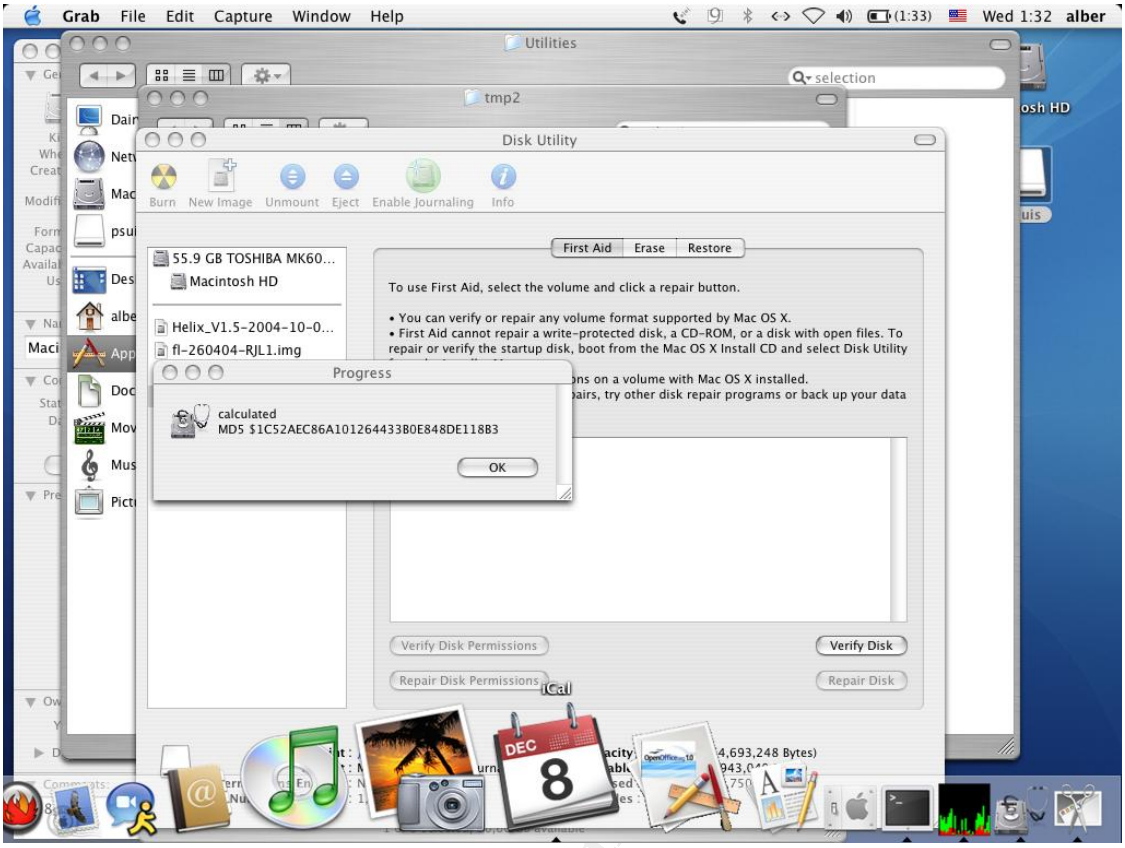

Info: psuis

Name : psuis
Type : Volume

Disk Identifier : disk2s2
Mount Point : /Volumes/psuis
File System : Mac OS Extended (Journaled)
Connection Bus : Disk Image
IO Content : Apple_HFS
Writable : No
Capacity : 1,2 GB (1.264.693.248 Bytes)
Free Space : 40,0 MB (41.943.040 Bytes)
Used : 1,1 GB (1.222.750.208 Bytes)
Number of Files : 7.970
Number of Folders : 1.215
Permissions Enabled : No
Can Turn Permissions Off : Yes
Can Be Formatted : Yes
Bootable : Yes
Supports Journaling : Yes
Journaled : Yes
Disk Number : 2
Partition Number : 2

I now use the checksumming functionality from Disk Utility to produce an md5 checksum that I will use and compare once I have transferred the file to my forensic workstation. The initial result, as it can be seen in the next screenshot, is the following:

1C52AEC86A101264433B0E848DE118B3



To use First Aid, select the volume and click a repair button.

- You can verify or repair any volume format supported by Mac OS X.
- First Aid cannot repair a write-protected disk, a CD-ROM, or a disk with open files. To repair or verify the startup disk, boot from the Mac OS X Install CD and select Disk Utility

```
calculated
MD5 $1C52AEC86A101264433B0E848DE118B3
```

Once the image is created, I use netcat to transfer it to my forensic workstation. First I start the listener:

```
[root@LinuxForensics linux_x86_static]# nc -l -p 31329 > /tmp/forensics/mac_user.dd
```

And then I use dd and netcat in the Mac:

```
Dairy:/suis$ dd if=/tmp2/forensics/mac_user.dmg  | nc 192.168.2.1 31329
```

And I finally manage to transfer the image to my forensic workstation after almost half an hour:

```
2472722+0 records in
2472722+0 records out
1266033401 bytes transferred in 1667.230825 secs (759363 bytes/sec)
```

I perform an md5 checksum using the md5sum command and I check that it coincides with the md5 checksum I previously performed before transferring the image.

The next problem comes now, when I try to add this image to the case I already had opened in autopsy. Unfortunately, autopsy in Linux does not recognize Mac OS Extended (and Journaled) file system.

I investigate two different ways out. The first one is linking the image in autopsy using the raw format. I did that but then a very limited palette of tools can be used in autopsy, mainly a strings study and the subsequent keyword search. Everything related to timeline analysis, file system, metadata layer and file name layer cannot be used. What is then my best next option?

I then remembered the USB memory stick that I also ‘seized’ together with the laptop. According to what my friend told me, she used the memory stick as a means of data transport between her Mac and her Windows-based desktop at the office. This fact gave me an idea on the fly: it was highly probable that the file system used in the memory stick would be accepted by autopsy.

Although I would not expect to have file repositories in the memory stick such as Internet history files or system registry files, I decided to analyse this image disk in the search of valuable information to describe the IT habits of its owner.

I connected the memory stick in one of the USB ports of my forensic workstation and I added the following line in my /etc/fstab to mount the memory stick in a read only mode:

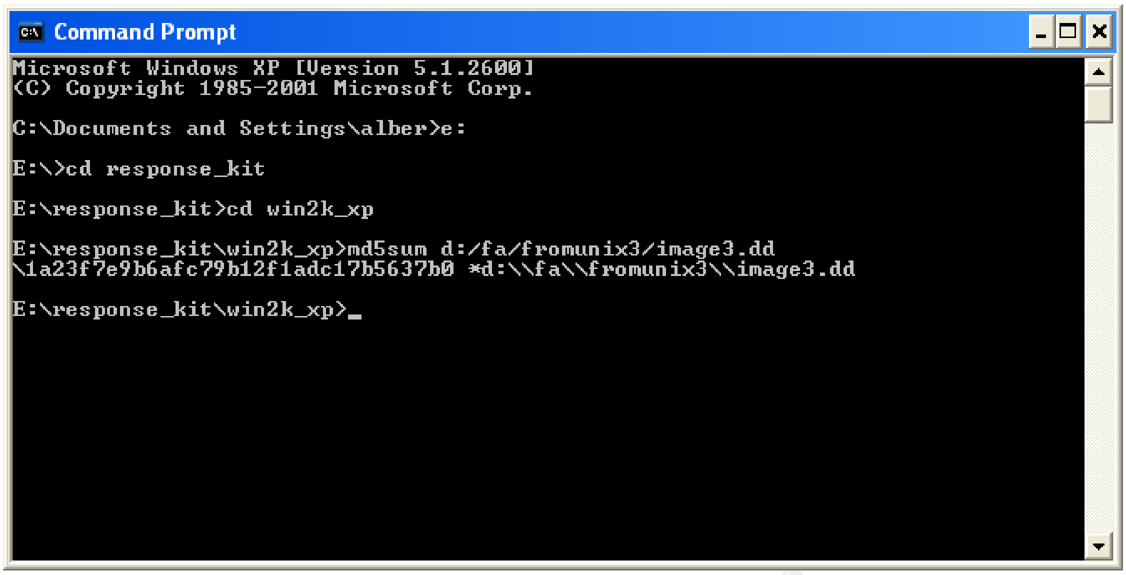
```
/dev/sda1            /mnt/flash            auto   noauto,ro     0 0
```

Next I used dd to create an image of this device:

```
[root@LinuxForensics forensics]# dd if=/dev/sda1 of=/tmp/forensics/image3.dd
```

I then linked this image to autopsy (it recognized it as a fat file system). This way, I will be able to perform a complete forensic analysis throughout all the different layers: file system layer, data layer, metadata layer, file name and finally timelines.

I also perform an md5 checksum of the USB memory stick image should I ever need to compare it with a new copy of the image to check its integrity (this time I do it on Windows) and I keep it in a safe place to guarantee continuity of possesion. It is important to remember that mistakes in a forensic investigation may corrupt evidence.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\alber>e:

E:\>cd response_kit

E:\response_kit>cd win2k_xp

E:\response_kit\win2k_xp>md5sum d:/fa/fromunix3/image3.dd
\1a23f7e9b6afc79b12f1adc17b5637b0 *d:\\fa\\fromunix3\\image3.dd

E:\response_kit\win2k_xp>_
```

2.5 Virus scanning of the system

Although this step is often skipped in forensics investigations, it is important to virus scan the image before even further analysing the system. I used Sophos Anti Virus version 3.88.0 but this does not mean any special preference for this brand. No known virus or malware was identified in the image.

2.6 Media analysis of the system

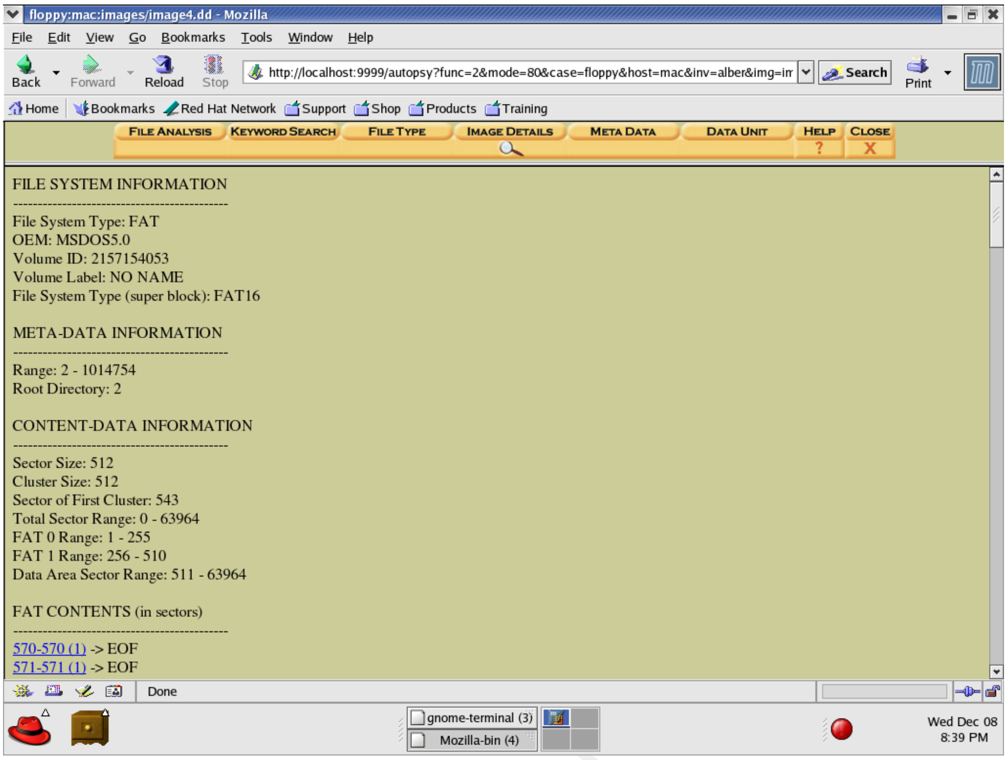
I will use the Sleuth Kit through the autopsy forensic browser version 1.75 to perform the media analysis of the system. As the Sleuth Kit mounts the to-be-analysed images in a read only mode, there is no possibility of modifying the evidence when performing the examination. Nevertheless, I made a copy of the image to be analysed and I stored it safely in a different media together with the checksum mentioned in the previous section.

The process I will follow is based on the different layers described in the Sleuth Kit web site (http://www.sleuthkit.org/sleuthkit/tools.php) and also reflected in books 2 and 3 of Track 8.

2.6.1 File system layer analysis

Once the image has been loaded into the autopsy browser as a FAT16 file system I click on details and a summary of all the information related to the file system , the meta data and the content data together with the contents of the file allocation table (FAT) as the next screen shot shows (no Master File Table, MFT as in NTFS or inodes as in Unix are used in FAT file systems).

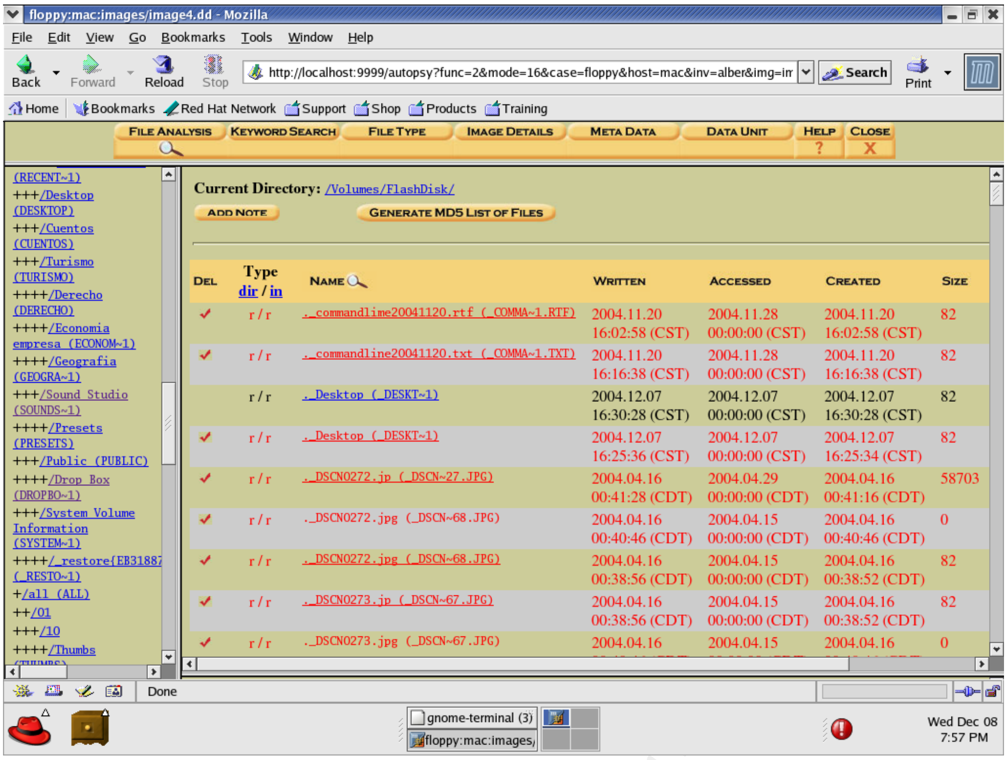
Using this autopsy functionality I reach the file allocation table contents. It is interesting to see how most of the files only occupy 1 sector and how one of the largest files is a Word document. The file allocation table provides me with the sector location information of the files stored in the disk when making the image.


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=80&case=floppy&host=mac&inv=alber&img=im
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
Done
gnome-terminal (3)
Mozilla-bin (4)
Wed Dec 08
8:39 PM

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT
OEM: MSDOS5.0
Volume ID: 2157154053
Volume Label: NO NAME
File System Type (super block): FAT16

META-DATA INFORMATION
--------------------------------------------
Range: 2 - 1014754
Root Directory: 2

CONTENT-DATA INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 512
Sector of First Cluster: 543
Total Sector Range: 0 - 63964
FAT 0 Range: 1 - 255
FAT 1 Range: 256 - 510
Data Area Sector Range: 511 - 63964

FAT CONTENTS (in sectors)
--------------------------------------------
570-570 (1) -> EOF
571-571 (1) -> EOF
```

2.6.2 File name layer analysis

I select the image and I press OK (important to remember this step, not really intuitive). The next screen tells me to select an analysis type. I first choose the file analysis.
It can be observed that the memory stick has been used quite heavily. There are plenty of folders, files and deleted folders and deleted files, as this screenshot shows:


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=16&case=floppy&host=mac&inv=alber&img=im
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
Done
gnome-terminal (3)
floppy:mac:images,
Wed Dec 08
7:57 PM

(RECENT~1)
+++/Desktop (DESKTOP)
+++/Cuentos (CUENTOS)
+++/Turismo (TURISMO)
++++/Derecho (DERECHO)
++++/Economia empresa (ECONOM~1)
++++/Geografia (GEOGRA~1)
+++/Sound Studio (SOUNDS~1)
++++/Presets (PRESETS)
+++/Public (PUBLIC)
++++/Drop Box (DROPBO~1)
+++/System Volume Information (SYSTEM~1)
++++/_restore{EB31887
(_RESTO~1)
+/all (ALL)
++/01
+++/10
++++/Thumbs

Current Directory: /Volumes/FlashDisk/

ADD NOTE GENERATE MD5 LIST OF FILES

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE
✓	r / r	._commandlime20041120.rtf (_COMMA~1.RTF)	2004.11.20 16:02:58 (CST)	2004.11.28 00:00:00 (CST)	2004.11.20 16:02:58 (CST)	82
✓	r / r	._commandline20041120.txt (_COMMA~1.TXT)	2004.11.20 16:16:38 (CST)	2004.11.28 00:00:00 (CST)	2004.11.20 16:16:38 (CST)	82
	r / r	._Desktop (_DESKT~1)	2004.12.07 16:30:28 (CST)	2004.12.07 00:00:00 (CST)	2004.12.07 16:30:28 (CST)	82
✓	r / r	._Desktop (_DESKT~1)	2004.12.07 16:25:36 (CST)	2004.12.07 00:00:00 (CST)	2004.12.07 16:25:34 (CST)	82
✓	r / r	._DSCN0272.jp (_DSCN~27.JPG)	2004.04.16 00:41:28 (CDT)	2004.04.29 00:00:00 (CDT)	2004.04.16 00:41:16 (CDT)	58703
✓	r / r	._DSCN0272.jpg (_DSCN~68.JPG)	2004.04.16 00:40:46 (CDT)	2004.04.15 00:00:00 (CDT)	2004.04.16 00:40:46 (CDT)	0
✓	r / r	._DSCN0272.jpg (_DSCN~68.JPG)	2004.04.16 00:38:56 (CDT)	2004.04.15 00:00:00 (CDT)	2004.04.16 00:38:52 (CDT)	82
✓	r / r	._DSCN0273.jp (_DSCN~67.JPG)	2004.04.16 00:38:56 (CDT)	2004.04.15 00:00:00 (CDT)	2004.04.16 00:38:52 (CDT)	82
✓	r / r	._DSCN0273.jpg (_DSCN~67.JPG)	2004.04.16	2004.04.15	2004.04.16	0

As it is a FAT image, no specific focus is required in terms of UID and GID since all files appear with UID and GID 0 (no specific access rights assigned to files in a FAT file system).

It is also clear that one of the systems providing data to the memory stick was the Mac, since typical Mac OS folders such as 'Drop Box' can be identified in the image.

The most common types of files that can be found in the image are:

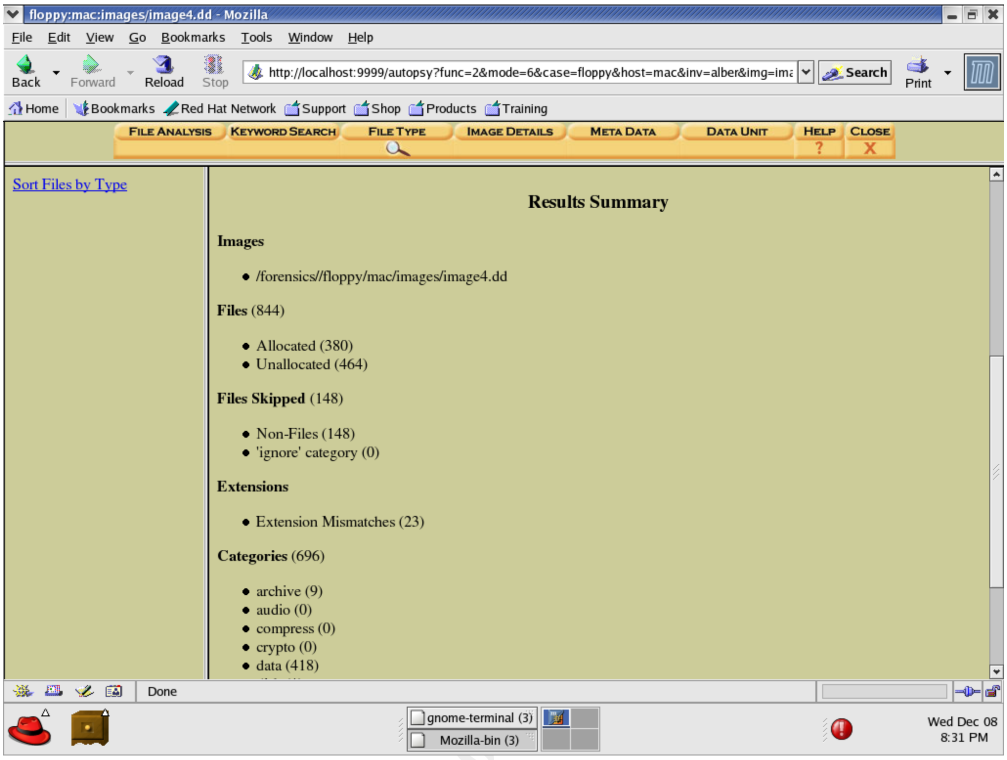
- Image files (.jpg): She stored plenty of digital images in the memory stick, probably coming from a Nikon digital camera, which normally uses file names following the naming convention of DSCN+number.

- Text-related files (.rtf, .txt, .doc and .pdf). Plenty of word processor documents have resided for some time in the memory stick.

- Presentation files (.ppt and .key): She has even used KeyNote (a presentation-making software from Apple). I can see a document with the extension .key together with plenty of .ppt files.

2.6.3 File type analysis

Using sorter through the autopsy browser I see that 844 files have been potentially identified. The file types that I already commented, 57 documents, 74 images and 80 text files are no surprise. However, 148 files were skipped and there are 23 extension mismatches. Eventually, I have to consider this data only as an initial estimation of what was really contained in this image.


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=6&case=floppy&host=mac&inv=alber&img=ima
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
Sort Files by Type
Done
gnome-terminal (3)
Mozilla-bin (3)
Wed Dec 08
8:31 PM

Results Summary

Images

- /forensics//floppy/mac/images/image4.dd

Files (844)

- Allocated (380)
- Unallocated (464)

Files Skipped (148)

- Non-Files (148)
- 'ignore' category (0)

Extensions

- Extension Mismatches (23)

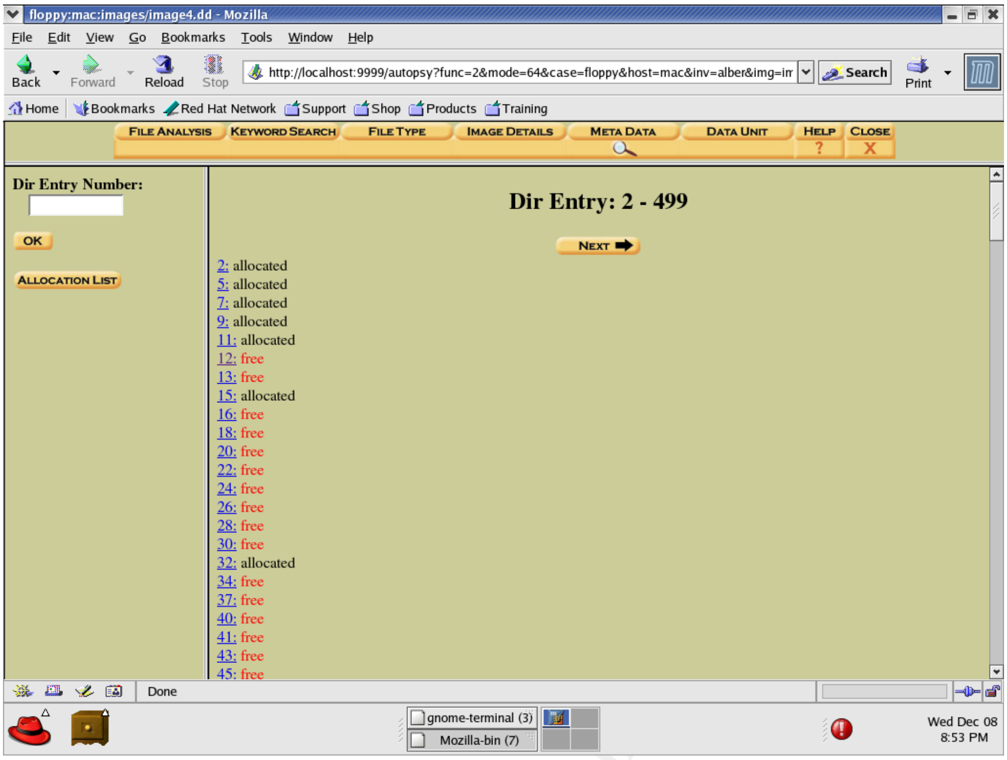
Categories (696)

- archive (9)
- audio (0)
- compress (0)
- crypto (0)
- data (418)

2.6.4 Meta data and data unit layer analysis

The meta data layer contains information about where the data is stored in the disk. Once the sectors a file occupies are known, it is only a question of going to the data unit section of the browser and analyzing the contents of the different data units. This meta data analysis gives also information about where the slack space (unused space) is present in the image.

Depending on the number and size of files that have been stored and deleted in the image, once the original sector numbers that a file was occupying are known, there is a possibility to recover the content of a deleted file. It only depends on whether, later on, another file was stored in the same system that actually would occupy some of the sectors of the previous file.


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=64&case=floppy&host=mac&inv=alber&img=im
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
Dir Entry Number:
OK
ALLOCATION LIST
NEXT
Done
gnome-terminal (3)
Mozilla-bin (7)
Wed Dec 08
8:53 PM

Dir Entry: 2 - 499

2: allocated
5: allocated
7: allocated
9: allocated
11: allocated
12: free
13: free
15: allocated
16: free
18: free
20: free
22: free
24: free
26: free
28: free
30: free
32: allocated
34: free
37: free
40: free
41: free
43: free
45: free

2.7 Timeline analysis

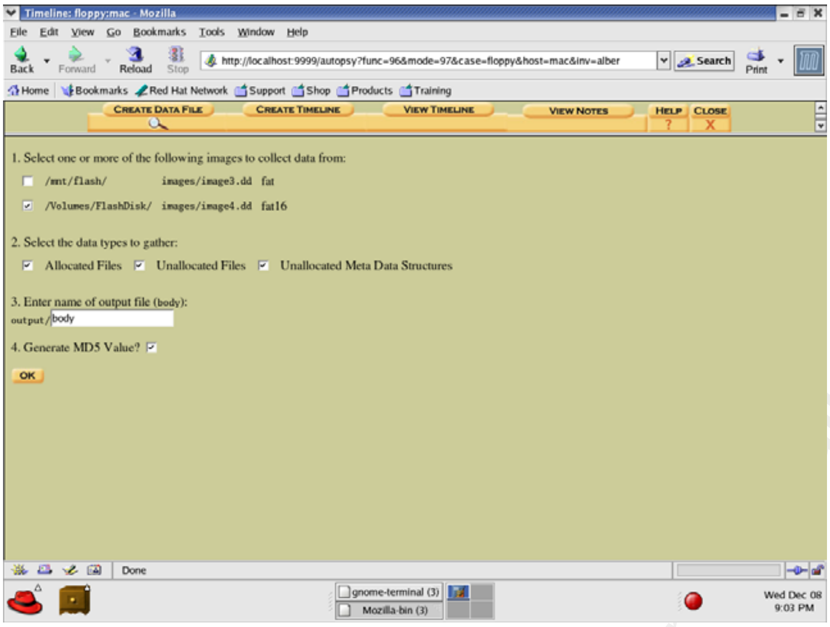
According to the help section in autopsy, a FAT file system has the following time information:

** Written: When the file was last written to. It is the ONLY required time in the FAT file system.*
** Accessed: When the file was last accessed. In FAT, it is only accurate to the day (not minute). It is an optional value, so some Operating Systems may not update it.*
** Created: When the file was created. It is also optional, so some Operating Systems may not update it. In fact, many Windows installations have a C-Time of 0 for directories such as C:\\Windows and C:\\Program Files.*

In general terms, a mac analysis uses ‘m’ as modified (the content of a file was modified), ‘a’ as accessed (the content of a file was accessed) and ‘c’ as changed (the file changed in ownership and attributes, normally it coincides with the creation time).

Using autopsy, I first create the data file and then I create the timeline. It can be seen in the next screenshots.

Data file creation:



http://localhost:9999/autopsy?func=96&mode=97&case=floppy&host=mac&inv=alber

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

□ /mnt/flash/ images/image3.dd fat

☑ /Volumes/FlashDisk/ images/image4.dd fat16

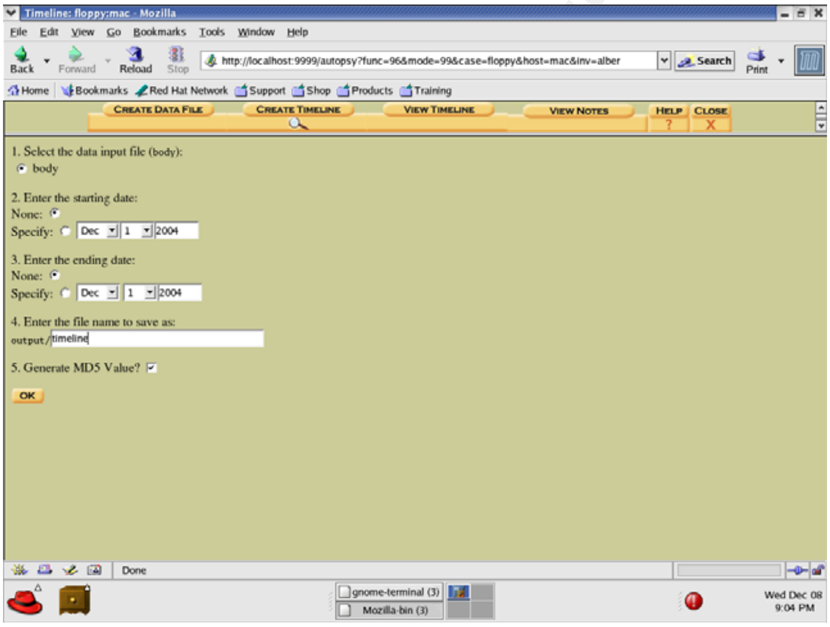
2. Select the data types to gather:

☑ Allocated Files ☑ Unallocated Files ☑ Unallocated Meta Data Structures

3. Enter name of output file (body):
output/body

4. Generate MD5 Value? ☑

Timeline creation:



http://localhost:9999/autopsy?func=96&mode=99&case=floppy&host=mac&inv=alber

1. Select the data input file (body):
● body

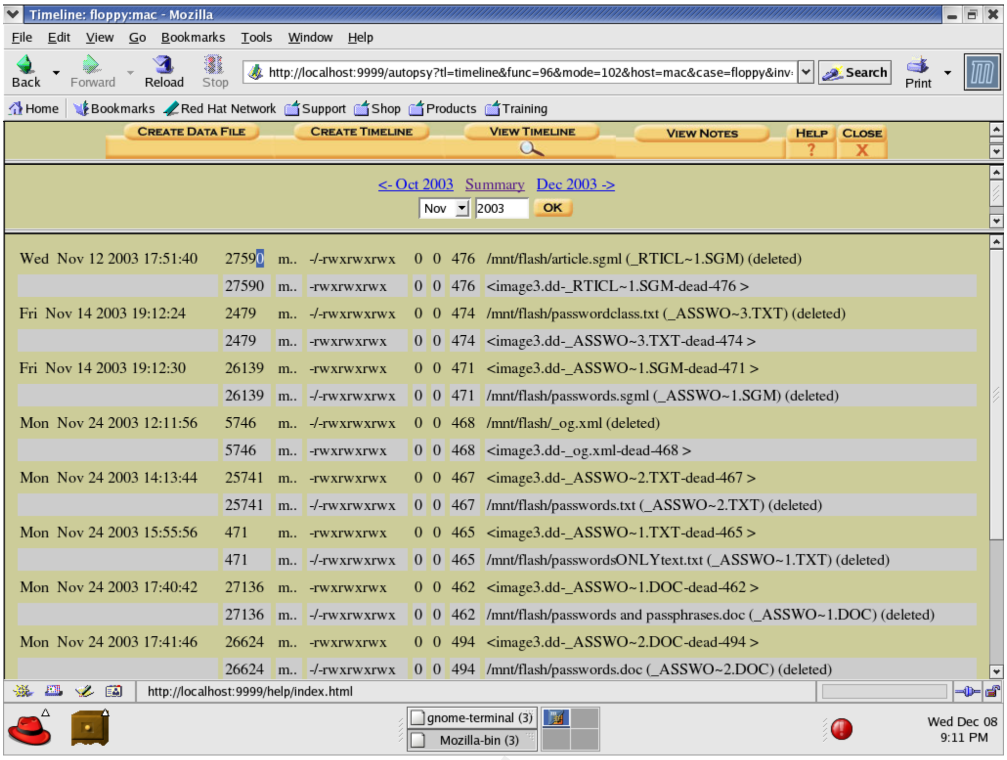
2. Enter the starting date:
None: ●
Specify: ○ Dec 1 2004

3. Enter the ending date:
None: ●
Specify: ○ Dec 1 2004

4. Enter the file name to save as:
output/timeline

5. Generate MD5 Value? ☑

The actual timeline information can be better studied outside autopsy using any text processor. Some interesting file names can be seen, such as e.g. passwordsONLYtext.txt or passwords and passphrases.doc. Depending on the scenario, these could very well be valuable information. I observe that the months of November 2004 and December 2004 were the most active months in terms of file creation and deletion. I could also see that she started to use the stick (accessed files) in January 2004.


Timeline: floppy:mac - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?tl=timeline&func=96&mode=102&host=mac&case=floppy&inv
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
CREATE DATA FILE
CREATE TIMELINE
VIEW TIMELINE
VIEW NOTES
HELP
CLOSE
<- Oct 2003 Summary Dec 2003 ->
Nov 2003 OK
http://localhost:9999/help/index.html
gnome-terminal (3)
Mozilla-bin (3)
Wed Dec 08
9:11 PM

```
Wed  Nov 12 2003 17:51:40   27590  m..  -/-rwxrwxrwx  0 0 476 /mnt/flash/article.sgml (_RTICL~1.SGM) (deleted)
                            27590  m..  -rwxrwxrwx    0 0 476 <image3.dd-_RTICL~1.SGM-dead-476 >
Fri  Nov 14 2003 19:12:24   2479   m..  -/-rwxrwxrwx  0 0 474 /mnt/flash/passwordclass.txt (_ASSWO~3.TXT) (deleted)
                            2479   m..  -rwxrwxrwx    0 0 474 <image3.dd-_ASSWO~3.TXT-dead-474 >
Fri  Nov 14 2003 19:12:30   26139  m..  -rwxrwxrwx    0 0 471 <image3.dd-_ASSWO~1.SGM-dead-471 >
                            26139  m..  -/-rwxrwxrwx  0 0 471 /mnt/flash/passwords.sgml (_ASSWO~1.SGM) (deleted)
Mon  Nov 24 2003 12:11:56   5746   m..  -/-rwxrwxrwx  0 0 468 /mnt/flash/_og.xml (deleted)
                            5746   m..  -rwxrwxrwx    0 0 468 <image3.dd-_og.xml-dead-468 >
Mon  Nov 24 2003 14:13:44   25741  m..  -rwxrwxrwx    0 0 467 <image3.dd-_ASSWO~2.TXT-dead-467 >
                            25741  m..  -/-rwxrwxrwx  0 0 467 /mnt/flash/passwords.txt (_ASSWO~2.TXT) (deleted)
Mon  Nov 24 2003 15:55:56   471    m..  -rwxrwxrwx    0 0 465 <image3.dd-_ASSWO~1.TXT-dead-465 >
                            471    m..  -/-rwxrwxrwx  0 0 465 /mnt/flash/passwordsONLYtext.txt (_ASSWO~1.TXT) (deleted)
Mon  Nov 24 2003 17:40:42   27136  m..  -rwxrwxrwx    0 0 462 <image3.dd-_ASSWO~1.DOC-dead-462 >
                            27136  m..  -/-rwxrwxrwx  0 0 462 /mnt/flash/passwords and passphrases.doc (_ASSWO~1.DOC) (deleted)
Mon  Nov 24 2003 17:41:46   26624  m..  -rwxrwxrwx    0 0 494 <image3.dd-_ASSWO~2.DOC-dead-494 >
                            26624  m..  -/-rwxrwxrwx  0 0 494 /mnt/flash/passwords.doc (_ASSWO~2.DOC) (deleted)
```

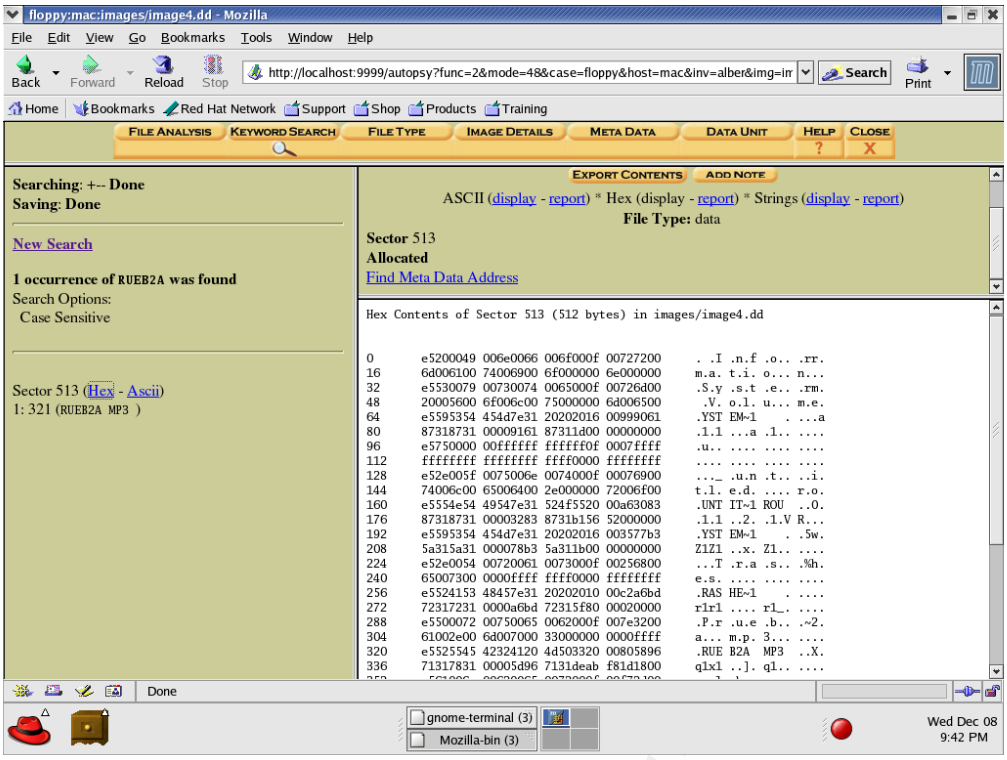
2.8 Recover a deleted file

Apparently an mp3 file was stored in the memory on 17 November 2004 and probably deleted on 24 November 2004 (because this name, PRUEB2A.mp3, does not appear anymore in the time line.

Wed Nov 17 2004 18:50:48 1580536 ..c -/-rwxrwxrwx 0 0 45 /mnt/flash/Prueb2a.mp3 (_RUEB2A.MP3) (deleted)
1580536 ..c -rwxrwxrwx 0 0 45 <image3.dd-_RUEB2A.MP3-dead-45 >
Wed Nov 17 2004 18:50:58 1580536 m.. -rwxrwxrwx 0 0 45 <image3.dd-_RUEB2A.MP3-dead-45 >
1580536 m.. -/-rwxrwxrwx 0 0 45 /mnt/flash/Prueb2a.mp3 (_RUEB2A.MP3) (deleted)

Wed Nov 24 2004 00:00:00 1580536 .a. -rwxrwxrwx 0 0 45 <image3.dd-_RUEB2A.MP3-dead-45 >
1580536 .a. -/-rwxrwxrwx 0 0 45 /mnt/flash/Prueb2a.mp3 (_RUEB2A.MP3) (deleted)

Making a string search of RUEB2A I get the sector in which the file was partly stored:


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
EXPORT CONTENTS ADD NOTE
Done
gnome-terminal (3)
Mozilla-bin (3)
Wed Dec 08
9:42 PM

http://localhost:9999/autopsy?func=2&mode=4&&case=floppy&host=mac&inv=alber&img=im

Searching: +-- **Done**
Saving: **Done**

New Search

1 occurrence of RUEB2A was found
Search Options:
Case Sensitive

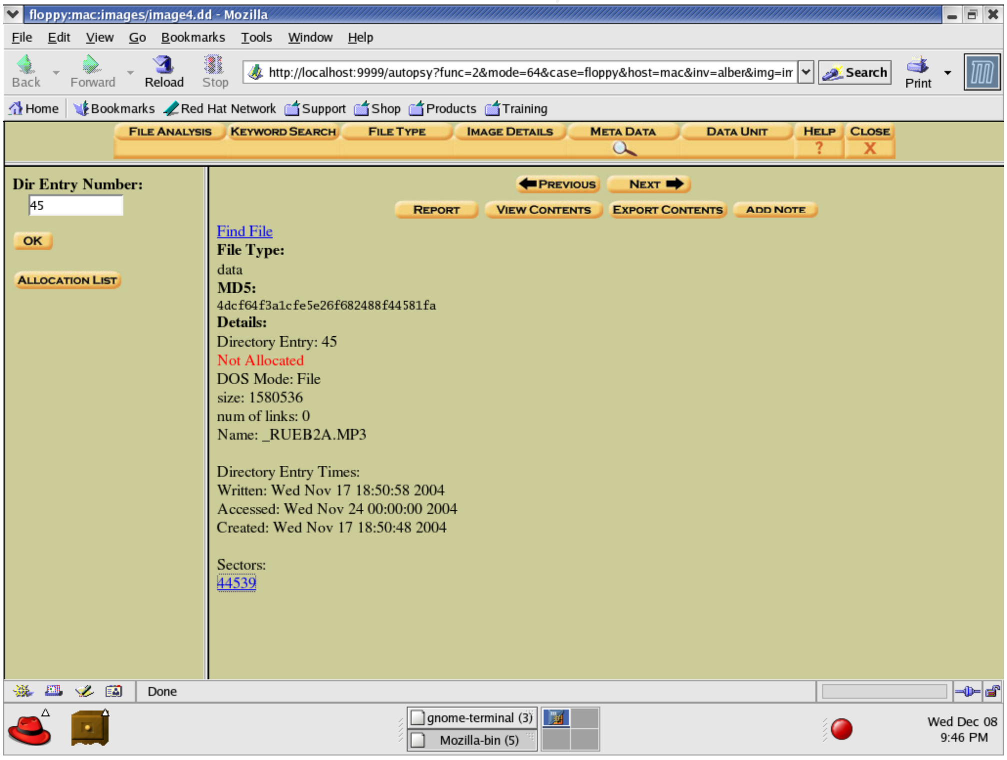
Sector 513 (Hex - Ascii)
1: 321 (RUEB2A MP3)

ASCII (display - report) * Hex (display - report) * Strings (display - report)
File Type: data

Sector 513
Allocated
Find Meta Data Address

```
Hex Contents of Sector 513 (512 bytes) in images/image4.dd


0       e5200049 006e0066 006f000f 00727200     . .I .n.f .o.. .rr.
16      6d006100 74006900 6f000000 6e000000     m.a. t.i. o... n...
32      e5530079 00730074 0065000f 00726d00     .S.y .s.t .e.. .rm.
48      20005600 6f006c00 75000000 6d006500      .V. o.l. u... m.e.
64      e5595354 454d7e31 20202016 00999061     .YST EM~1    . ...a
80      87318731 00009161 87311d00 00000000     .1.1 ...a .1.. ....
96      e5750000 00ffffff ffffff0f 0007ffff     .u.. .... .... ....
112     ffffffff ffffffff ffff0000 ffffffff     .... .... .... ....
128     e52e005f 0075006e 0074000f 00076900     ..._ .u.n .t.. ..i.
144     74006c00 65006400 2e000000 72006f00     t.l. e.d. .... r.o.
160     e5554e54 49547e31 524f5520 00a63083     .UNT IT~1 ROU  ..0.
176     87318731 00003283 8731b156 52000000     .1.1 ..2. .1.V R...
192     e5595354 454d7e31 20202016 003577b3     .YST EM~1    . .5w.
208     5a315a31 000078b3 5a311b00 00000000     Z1Z1 ..x. Z1.. ....
224     e52e0054 00720061 0073000f 00256800     ...T .r.a .s.. .%h.
240     65007300 0000ffff ffff0000 ffffffff     e.s. .... .... ....
256     e5524153 48457e31 20202010 00c2a6bd     .RAS HE~1    . ....
272     72317231 0000a6bd 72315f80 00020000     r1r1 .... r1_. ....
288     e5500072 00750065 0062000f 007e3200     .P.r .u.e .b.. .~2.
304     61002e00 6d007000 33000000 0000ffff     a... m.p. 3... ....
320     e5525545 42324120 4d503320 00805896     .RUE B2A  MP3  ..X.
336     71317831 00005d96 7131deab f81d1800     q1x1 ..]. q1.. ....
```

But I am not able to get the whole file back because some of its sectors have been used to stored other file data, as I was already explaining in the meta data layer analysis section. E. g. this case only 1 sector is mentioned in autopsy.


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
OK
ALLOCATION LIST
PREVIOUS NEXT
REPORT VIEW CONTENTS EXPORT CONTENTS ADD NOTE
Done
gnome-terminal (3)
Mozilla-bin (5)
Wed Dec 08
9:46 PM

http://localhost:9999/autopsy?func=2&mode=6&&case=floppy&host=mac&inv=alber&img=im

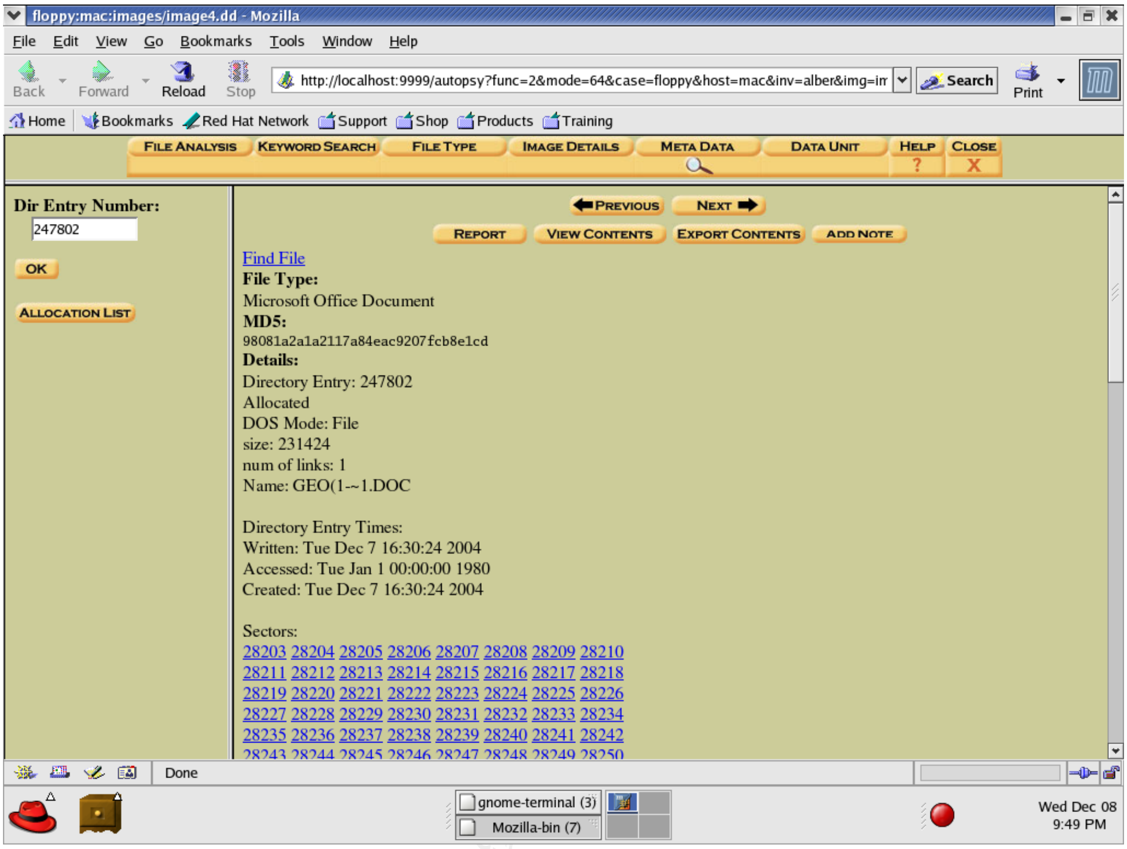
Dir Entry Number:
45

Find File
File Type:
data
MD5:
4dcf64f3a1cfe5e26f682488f44581fa
Details:
Directory Entry: 45
Not Allocated
DOS Mode: File
size: 1580536
num of links: 0
Name: _RUEB2A.MP3

Directory Entry Times:
Written: Wed Nov 17 18:50:58 2004
Accessed: Wed Nov 24 00:00:00 2004
Created: Wed Nov 17 18:50:48 2004

Sectors:
44539

A non-deleted file such us the .doc file showed in the previous screenshot occupies, as it can be seen several sectors (they are available just by clicking on them).


floppy:mac:images/image4.dd - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:9999/autopsy?func=2&mode=64&case=floppy&host=mac&inv=alber&img=im
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
FILE ANALYSIS KEYWORD SEARCH FILE TYPE IMAGE DETAILS META DATA DATA UNIT HELP CLOSE
?
X
PREVIOUS NEXT
REPORT VIEW CONTENTS EXPORT CONTENTS ADD NOTE
OK
ALLOCATION LIST
Done
gnome-terminal (3)
Mozilla-bin (7)
Wed Dec 08
9:49 PM

Dir Entry Number:
247802

Find File
File Type:
Microsoft Office Document
MD5:
98081a2a1a2117a84eac9207fcb8e1cd
Details:
Directory Entry: 247802
Allocated
DOS Mode: File
size: 231424
num of links: 1
Name: GEO(1-~1.DOC

Directory Entry Times:
Written: Tue Dec 7 16:30:24 2004
Accessed: Tue Jan 1 00:00:00 1980
Created: Tue Dec 7 16:30:24 2004

Sectors:
28203 28204 28205 28206 28207 28208 28209 28210
28211 28212 28213 28214 28215 28216 28217 28218
28219 28220 28221 28222 28223 28224 28225 28226
28227 28228 28229 28230 28231 28232 28233 28234
28235 28236 28237 28238 28239 28240 28241 28242
28243 28244 28245 28246 28247 28248 28249 28250

2.9 *String search*

The keyword search is always fun. I start with an unexpected word: security. Interestingly enough, 307 occurrences of security were found in the image. A pretty high number considering the use of this memory stick as a mere data transport.

As a second try I used the work hacking and no occurrences appear. I also perform a search of IP addresses and I get 242 occurrences found in the image.

I recommend reading the section in autopsy about what can and cannot be found using strings. It is clear then than the results provided by autopsy can only be a subset of the real number of occurrences of a word or a regular expression in an image.

The type of words I could look for in the image depends very much on the scenario where the forensic analyst is called in. For example, many cool rootkits are coming from Romania, I could then look for Romanian words. In this case, I just leave this section here since, although I could very well try with thousands of other words and regular expressions, it is always the same process to perform these searches.

2.10 Conclusions

Based on the previous comprehensive analysis, this USB memory stick has been heavily used specially in the last year as a data bridge between an iBook and a Windows workstation. Mostly the files that were transferred were text-related files and digital photos. Apparently a possible content of the files could be security-related due to the high number of occurrences of the words security and passwords. No major and visible hacking tools or programs were initially discovered in the system.

From the analysed data, I conclude that it is a typical system from an average nowadays' user who uses word-processor tools, produces digital photo files and uses removable media to transport data from her private laptop to her office desktop.

Probably the reason why no malware or virus coming from Internet was identified in the image is due to the following reasons:

- At home, she connects to the Internet using her iBook and I could check that the personal firewall present in Mac OS X was activated with pretty strict rules closing all native services (such as personal file sharing, Windows file sharing, personal web sharing, remote login, FTP access, remote Apple events and printer sharing).

- In her company, a middle-sized PR company, all staff connect to the Internet via a web proxy that makes HTTP and HTTPS virus and malware filtering.

I decided to present this scenario because I estimate it is currently becoming a very common case information security groups in most companies have to deal with: staff transferring large amount of data thanks to the USB memory sticks. And also because everybody loves stories, and juries too.

As a general conclusion, I recommend companies to elaborate a simple but effective policy about the use of removable storage media on their desktops (together maybe with a forensics policy). Whatever the policy finally agreed, at least all players will have a clear picture of what is allowed and what is not.

I also draw some additional generic conclusions:

- Every forensic analysts need to follow a formal investigation methodology. This will make a difference where presenting evidence in front of a jury.

- Describing the different hardware components can be a difficult task especially investigating seized laptops. Most of the hardware elements are embedded in the laptop case and getting to know the brand and serial number of e.g. the hard disk can be difficult.

- The greatest challenge to overcome when doing real-life forensic analysis is usually the size of the image the investigator has to deal with. Although in our two examples in this assignment the size of the studied images has been limited due to practical reasons, in real life forensic analysts have to deal with images that have several gigabytes. This fact increases tremendously the complexity of the analysis work in terms of time and filters the investigator has to apply to reach clear conclusions.

- Clearly documenting each and every step of the investigation is of the utmost importance. It is the way to be able to reproduce the analysis and to justify in front of a jury all the actions performed in a system. John Green, my SANS Track 8 instructor suggested the idea of having a record taker, a junior colleague willing to take notes and learn forensics. It is a fact that it normally takes from 12 to 18 months for a case to reach a court.

- It was also interesting the experience gained with the different file systems present in this study scenario. Mac OS extended file system (HFS plus) is still a proprietary file system that The Sleuth Kit cannot mount yet. FAT file systems, on the contrary, can easily be mounted by the Sleuth Kit.

My friend got her iBook laptop back ‘safe and sound’ after my investigations.

3. Annex

3.1 Timeline file of the floppy image in part 1

see

```
Sat Feb 03 2001 19:44:16
36864 m.. -rwxrwxrwx 0 0 5 <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>
36864 m.. -/-rwxrwxrwx 0 0 5  a:\/CamShell.dll (_AMSHELL.DLL) (deleted)

Thu Apr 22 2004 17:31:06
32256 m.. -/-rwxrwxrwx 0  0  13  a:\/Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
33423 m.. -/-rwxrwxrwx 0  0  17  a:\/Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)

Fri Apr 23 2004 11:53:56      727 m.. -rwxrwxrwx 0  0  28  <fl-260404-RJL1.img-
_ndex.htm-dead-28>
       727 m.. -/-rwxrwxrwx 0  0  28  a:\/_ndex.htm (deleted)

Fri Apr 23 2004 12:54:32   215895 m.. -/-rwxrwxrwx 0  0  23
a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)

Fri Apr 23 2004 12:55:26   307935 m.. -/-rwxrwxrwx 0  0  20
a:\/Password_Policy.doc (PASSWO~1.DOC)

Fri Apr 23 2004 15:10:50    22528 m.. -/-rwxrwxrwx 0  0  27
a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)

Fri Apr 23 2004 15:11:10    42496 m.. -/-rwxrwxrwx 0  0  9
a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)

Mon Apr 26 2004 01:00:00    32256 .a. -/-rwxrwxrwx 0  0  13
a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
       727 .a. -/-rwxrwxrwx 0  0  28  a:\/_ndex.htm (deleted)
       727 .a. -rwxrwxrwx 0  0  28  <fl-260404-RJL1.img-_ndex.htm-dead-28>
     36864 .a. -rwxrwxrwx 0  0  5  <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>
     36864 .a. -/-rwxrwxrwx 0  0  5  a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
     22528 .a. -/-rwxrwxrwx 0  0  27  a:\/Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
    215895 .a. -/-rwxrwxrwx 0  0  23  a:\/Remote_Access_Policy.doc
(REMOTE~1.DOC)
     33423 .a. -/-rwxrwxrwx 0  0  17  a:\/Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
    307935 .a. -/-rwxrwxrwx 0  0  20  a:\/Password_Policy.doc (PASSWO~1.DOC)
     42496 .a. -/-rwxrwxrwx 0  0  9  a:\/Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
```

```
Mon Apr 26 2004 10:46:18    36864 ..c -/-rwxrwxrwx 0  0  5  a:\/CamShell.dll
(_AMSHELL.DLL) (deleted)
    36864 ..c -rwxrwxrwx 0  0  5  <fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>

Mon Apr 26 2004 10:46:20    42496 ..c -/-rwxrwxrwx 0  0  9
a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)

Mon Apr 26 2004 10:46:22    32256 ..c -/-rwxrwxrwx 0  0  13
a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)

Mon Apr 26 2004 10:46:24    33423 ..c -/-rwxrwxrwx 0  0  17
a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)

Mon Apr 26 2004 10:46:26   307935 ..c -/-rwxrwxrwx 0  0  20
a:\/Password_Policy.doc (PASSWO~1.DOC)

Mon Apr 26 2004 10:46:36   215895 ..c -/-rwxrwxrwx 0  0  23
a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)

Mon Apr 26 2004 10:46:44    22528 ..c -/-rwxrwxrwx 0  0  27
a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)

Mon Apr 26 2004 10:47:36      727 ..c -/-rwxrwxrwx 0  0  28  a:\/_ndex.htm
(deleted)
      727 ..c -rwxrwxrwx 0  0  28  <fl-260404-RJL1.img-_ndex.htm-dead-28>
```

3.2 Complete autopsy file information set from floppy image in part 1

```
0|a:\/CamShell.dll (_AMSHELL.DLL) (deleted)|0|5|33279|-/-
rwxrwxrwx|0|0|0|0|36864|1082959200|981251056|1082994378|512|0
0|a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)|0|9|33279|-/-
rwxrwxrwx|1|0|0|0|42496|1082959200|1082751070|1082994380|512|0
0|a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)|0|13|33279|-/-
rwxrwxrwx|1|0|0|0|32256|1082959200|1082673066|1082994382|512|0
0|a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)|0|17|33279|-/-
rwxrwxrwx|1|0|0|0|33423|1082959200|1082673066|1082994384|512|0
0|a:\/Password_Policy.doc (PASSWO~1.DOC)|0|20|33279|-/-
rwxrwxrwx|1|0|0|0|307935|1082959200|1082742926|1082994386|512|0
0|a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)|0|23|33279|-/-
rwxrwxrwx|1|0|0|0|215895|1082959200|1082742872|1082994396|512|0
0|a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)|0|27|33279|-/-
rwxrwxrwx|1|0|0|0|22528|1082959200|1082751050|1082994404|512|0
0|a:\/_ndex.htm (deleted)|0|28|33279|-/-
rwxrwxrwx|0|0|0|0|727|1082959200|1082739236|1082994456|512|0
class|host|start_time
```

```
body|LinuxForensics|1102331268
md5|file|st_dev|st_ino|st_mode|st_ls|st_nlink|st_uid|st_gid|st_rdev|st_size|st_atim
e|st_mtime|st_ctime|st_blksize|st_blocks
0|<fl-260404-RJL1.img-_AMSHELL.DLL-dead-5>|0|5|33279|-
rwxrwxrwx|0|0|0|0|36864|1082959200|981251056|1082994378|512|0
0|<fl-260404-RJL1.img-_ndex.htm-dead-28>|0|28|33279|-
rwxrwxrwx|0|0|0|0|727|1082959200|1082739236|1082994456|512|0
```

3.3 Strings output of CamShell.dll in Part 1

```
ll\SheCamouflageShell
ShellExt
VB5!
CamShell
BitmapShellMenu
CamouflageShell
CamouflageShell
Shell_Declares
Shell_Functions
ShellExt
modShellRegistry
kernel32
lstrcpyA
lstrlenA
ole32.dll
CLSIDFromProgID
StringFromGUID2
ReleaseStgMedium
shell32.dll
DragQueryFileA
RtlMoveMemory
VirtualProtect
gdi32
CreateICA
GetTextMetricsA
CreateCompatibleDC
DeleteDC
GetObjectA
CreateBitmapIndirect
SelectObject
StretchBlt
DeleteObject
FindResourceA
advapi32.dll
user32
LoadBitmapA
LoadResource
advapi32
RegQueryValueExA
ModifyMenuA
InsertMenuA
SetMenuItemBitmaps
LoadLibraryA
SystemParametersInfoA
GetFullPathNameA
RegOpenKeyExA
RegCloseKey
__vbaI4Var
VBA6.DLL
__vbaCopyBytes
__vbaFreeStrList
__vbaFreeObj
__vbaCastObj
__vbaLateIdCallLd
__vbaHresultCheckObj
__vbaI2I4
__vbaNew2
7__vbaObjSet
__vbaStrCmp
__vbaStrVarVal
IContextMenu_QueryContextMenu
__vbaBoolVar
__vbaObjSetAddref
__vbaAptOffset
__vbaAryDestruct
IShellExtInit_Initialize
__vbaStrVarCopy
__vbaAryUnlock
__vbaGenerateBoundsError
__vbaAryLock
IContextMenu
__vbaStr2Vec
```

```
__vbaAryMove
__vbaStrCat
__vbaStrToUnicode
__vbaFreeVar
F__vbaStrVarMove
__vbaStrMove
__vbaStrCopy
__vbaErrorOverflow
__vbaFreeStr
__vbaSetSystemError
__vbaStrToAnsi
Class
C:\WINDOWS\SYSTEM\MSVBVM60
.DLL\3
VBRUN
FIShellExtInit
C:\My Documents\VB
Programs\Camouflage\Shell\IctxMen
u.tlb
IContextMenu_TLB
IContextMenu_GetCommandString
IContextMenu_InvokeCommand
__vbaRedim
__vbaUbound
__vbaVar2Vec
__vbaRecDestruct
__vbaLsetFixstr
__vbaLsetFixstrFree
__vbaLenBstr
__vbaFreeVarList
__vbaFixstrConstruct
__vbaVarTstEq
__vbaVarMove
__vbaVarCopy
__vbaVarDup
7m_szFile
IContextMenu
IShellExtInit
pidlFolder
lpdobj
hKeyProgID
hMenu
indexMenu
idCmdFirst
idCmdLast
uFlags
```

```
idCmd
pwReserved
pszName
cchMax
lpcmi
pVfk
pIVR
Pj@j
L$ j
7hd(
7hd(
7hd(
Sh|)
j4hl)
7PWh
Qh<)
Vh|)
j4hl)
WPQj
B4Ph(.
PQWWR
`SVW
Ph .
Ph .
Vh|)
Vh|)
Ph .
t        9u
PVQR
MSVBVM60.DLL
_CIcos
_adj_fptan
__vbaVarMove
__vbaFreeVar
__vbaAryMove
__vbaLenBstr
__vbaStrVarMove
__vbaAptOffset
__vbaFreeVarList
_adj_fdiv_m64
_adj_fprem1
__vbaCopyBytes
__vbaStrCat
__vbaLsetFixstr
__vbaRecDestruct
__vbaSetSystemError
```

```
__vbaHresultCheckObj
_adj_fdiv_m32
__vbaAryDestruct
EVENT_SINK2_Release
__vbaObjSet
_adj_fdiv_m16i
__vbaObjSetAddref
_adj_fdivr_m16i
__vbaBoolVar
_CIsin
__vbaChkstk
EVENT_SINK_AddRef
__vbaGenerateBoundsError
__vbaStrCmp
__vbaVarTstEq
__vbaI2I4
DllFunctionCall
_adj_fpatan
__vbaFixstrConstruct
__vbaLateIdCallLd
__vbaRedim
EVENT_SINK_Release
_CIsqrt
EVENT_SINK_QueryInterface
__vbaStr2Vec
__vbaExceptHandler
__vbaStrToUnicode
_adj_fprem
_adj_fdivr_m64
__vbaFPException
__vbaUbound
__vbaStrVarVal
__vbaLsetFixstrFree
_CIlog
__vbaErrorOverflow
__vbaVar2Vec
__vbaNew2
_adj_fdiv_m32i
_adj_fdivr_m32i
__vbaStrCopy
EVENT_SINK2_AddRef
__vbaFreeStrList
_adj_fdivr_m32
_adj_fdiv_r
__vbaI4Var
__vbaAryLock
__vbaVarDup
__vbaStrToAnsi
__vbaVarCopy
_CIatan
__vbaStrMove
__vbaCastObj
__vbaStrVarCopy
_allmul
_CItan
__vbaAryUnlock
_CIexp
__vbaFreeStr
__vbaFreeObj
CamShell.dll
DllCanUnloadNow
DllGetClassObject
DllRegisterServer
DllUnregisterServer
_|:cu
```

3.4 Screen shots of the hidden files in Part 1

Apart from the two hidden files already showed in the text in Part 1 of this assignment, here the reader can find two screenshots of the other two hidden files:

- Hydrocarbon%20fuel%20cell%20page2.jpg
- pem_fuelcell.gif

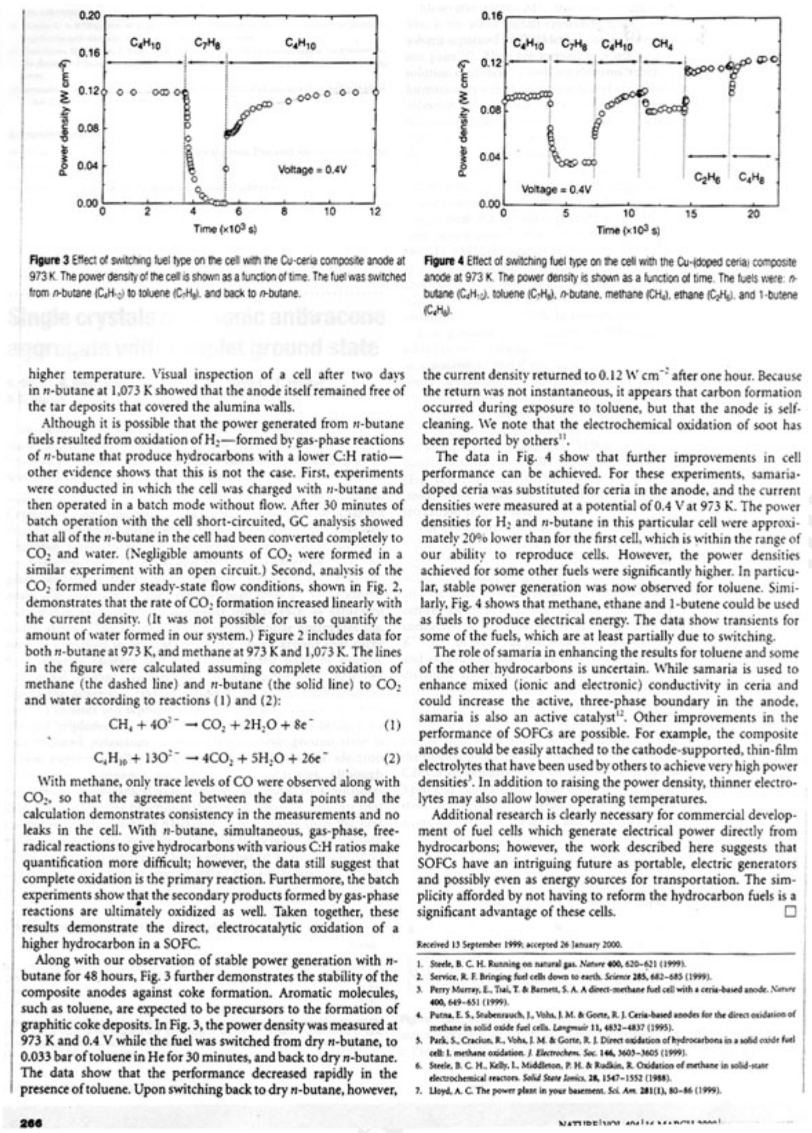

0.20
0.16
0.12
0.08
0.04
0.00
0
2
4
6
8
10
12
C_4H_{10}
C_7H_8
C_4H_{10}
Power density (W cm^{-2})
Time (×10^3 s)
Voltage = 0.4V

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.

0.16
0.12
0.08
0.04
0.00
0
5
10
15
20
C_4H_{10}
C_7H_8
C_4H_{10}
CH_4
C_2H_6
C_4H_8
Power density (W cm^{-2})
Time (×10^3 s)
Voltage = 0.4V

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):

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

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

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].

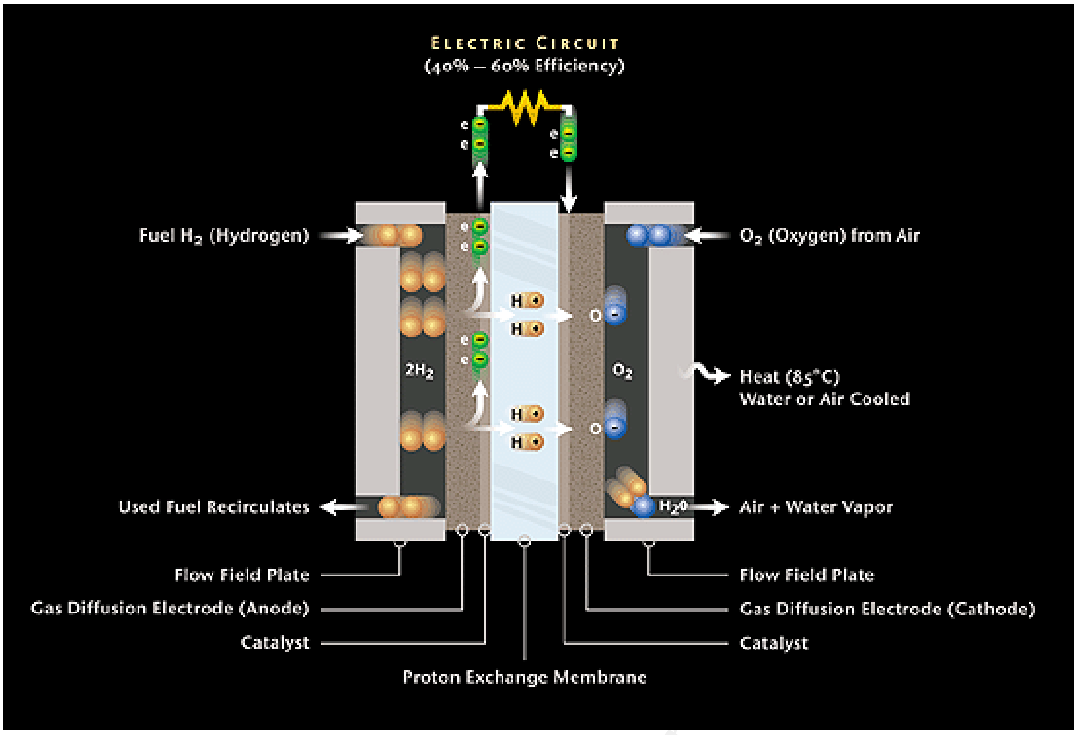
The data in Fig. 4 show that further improvements in cell performance can be achieved. For these experiments, samaria-doped ceria was substituted for ceria in the anode, and the current densities were measured at a potential of 0.4 V at 973 K. The power densities for H_2 and *n*-butane in this particular 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).



4. Administrivia reference

This assignment follows GIAC Certification Administrivia version 2.8a (revised July 2004)