GCFA
(GIAC Certified Forensic Analyst)
Practical version 1.5

Forensic analysis of a Windows 98 system

Abstract: This paper contains two sections. The first section is the analysis of an unknown image. This image was seized from a person leaving an R&D facility. The second section is the analysis of a Windows 98 system. This system was not found to be compromised.

GCFA Practical, Version 1.5 (April 30, 2004)

Jerry A. Shenk
Submitted: November 20, 2004

Table of Contents

Assignment 1 – Analyze an unknown image

Introduction:

Ballard Industries is a designer and manufacturer of fuel cells. Ballard has had an exclusive product and clientele for several years. Recently, many customers have not been re-ordering and follow-up calls indicate that they are now ordering from a competitor, Rift, Inc. During the course of the ensuing investigation, a floppy disk was carried out of the R&D lab by Robert Leszczynski on April 26, 2004. The act of carrying the disk out of the R&D lab is a violation of company policy so the disk was seized by the security guard on duty.

Executive summary:

It has been determined through forensic investigation that the floppy seized from Robert Leszczynski on April 26, 2004 at approximately 4:45 PM MST contains a partial copy of the “Client Authorization Table” database, images relating to fuel cells, a page from a technical document relating to fuel cells and a memo that suggests the desire of Robert J. Leszczynski to sell further information for a fee of 5 million (dollars assumed, currency not specified). These items have been included in Appendix A. Before we go into the details, it should be stressed that just because we’ve mentioned a particular name, doesn’t mean that he’s guilty…perhaps he’s being framed.

The information in Appendix A should be compared with Ballard proprietary documentation to determine if this is in fact proprietary information. From a forensic and evidentiary standpoint, it would appear that proprietary information has been stored on the floppy in question and was intended to be removed from the R&D lab. How the data got there cannot be known at this point since we have not identified nor examined the computer that this disk came from nor do we know for certain that this was actually Robert’s work. It is possible that this disk was planted in Robert’s briefcase to be retrieved later. It is also probable that more than one computer was involved in creating this floppy because of file timestamp discrepancies as detailed in the timeline analysis.

An immediate examination of computers in the R&D lab should begin. Other computers that Robert has access to should also be examined. This examination should target the existence of the Camouflage program and any traces of its existence through timeline analysis and analysis of unallocated space to find the evidence even if the files have been deleted. The existence of the Camouflage program can be determined by a camshell.dll, camouflage.exe, a camouflage directory in the “program folders” directory or the install file camou121.zip in either active or deleted files. The installation of the program also creates a Camouflage folder under Start, Program files which could be the easiest (although incomplete) thing to look for in a currently running system. The files references in Appendix A should also be key components of this investigation. The files listed in the directory of the floppy should be searched for also. These files are referenced in the Image Analysis section of this document but they are also listed in Appendix B so that they can be easily given to the investigators that work on the R&D computers.

As this is a high-level issue both criminally and from the business perspective of Ballard Industries, it is further recommended that a full forensic analysis be done of any computers that Mr. Leszczynski has access to. A search should be done on these computers for terms that are quite specific to this investigation including camshell, camouflaugeshell. See the “Image Analysis” section of this paper for reference to this search.

One additional factor that could help identify the computers that these files were generated on is that some of the files have a company name of Cisco Systems, Inc. That seems like an odd identifying mark and may be of value if it were searched for. Care should be taken with this search in particular as it is not specific to the current incident and may be related to a peculiarity of a more company-wide installation and configuration practice.

By analyzing the dates and times of certain file access (see “timeline analysis” section), it seems like the files were created or modified on April 22, 2004 at 4:31 PM and on April 23, 2004 between 10:53 and 2:11 PM. In addition to examining the computer systems in and around the R&D lab, there should be an examination of e-mail, phone records and physical access logs to determine if there is any corroborating evidence placing Mr. Leszczynski in the building, room or cubicle at these times. The building access logs would be quite valuable but if there is a record of accesses to specific rooms, that would be even better. Depending on the types of phone access, it may even be possible to identify that a Mr. Leszczynski used the phone beside a particular computer during these times. Even if there is only enough evidence to place him on the premises at these times, that will still help in identifying if he in fact is responsible for the content on these floppies.

It would be good to also check the original floppy for fingerprints. It is possible that somebody else placed this floppy in Mr. Leszczynski’s briefcase without his knowledge. Having fingerprints from the floppy and matching them to the area around his desk, phone and computer would also help in determining if he was actually involved. Just because his fingerprints are ABSENT from the floppy does not mean that he is not involved; perhaps he wore gloves. Likewise, if his fingerprints do exist on the floppy, that doesn’t necessarily prove that he’s the guilty party.

This examination should be kept as quiet as possible because we do not know who the involved parties are. It is possible that system administrators and/or security personal are involved so all investigation should be done in pairs with one person taking detailed notes and providing verification.

Forensic methodology overview

The evidence that we have at this point is a floppy disk that was seized from Robert Leszczynski on April 26, 2004 at 4:45 PM MST. The time information on this image points to customer and fuel cell data being collected on April 22 and 23 and being encrypted and hidden in policy documents. On the morning of April 26th, these files were copied to the seized floppy. This floppy was then taken from the R&D lab later that afternoon.

It seems evident based on the image analysis the floppy this image was taken from is a computer running a version of Windows although we do not have access to the computer this floppy was created with at this time.

Early in the investigation, we came up with a hypothesis that a program called Camouflage has been in use because of initial examination detail. Data in the floppy that was seized indicates that this program was used to hide data. That clearly fits within the stated suspicion that "…Rift, Inc. somehow has received proprietary information from Ballard…"

It is imperative that we as investigators maintain an open mind but it also seems prudent to dig deeper into researching the Camouflage program. The Coroner's Toolkit references an image[1] from the 1929 painting by René Magritte's entitled "The Treachery of Images". This reminds us that things are often not what they seem so we must constantly double-check our assumptions.

A string search of the floppy image yields a number of strings that support the fact that a Camouflage executable was stored on the seized floppy disk.

In this section, we will prove that the Camouflage program was used to encrypt and hide information inside text documents. The hidden information is in Appendix A.

There is a deleted windows dll in the image that suggests that Camouflage is the program that was used to encrypt the data. A little over 10% of that file was overwritten but the remaining 88% could be compared with the same fragment of a working copy of Camouflage to prove that these files are in fact the same. This working copy of Camouflage can be used to decrypt the data if the correct password is used. There is a flaw in the encryption program that enables the password to be recovered. Using this flaw, encrypted data can be extracted. By being able to extract the data, we are virtually assured that this was the program used to hide the data.

This report will go over the procedure used for examining the floppy image

[1] http://www.fish.com/tct/FAQ.html#pipe

including verification of the integrity of the data being analyzed at various stages of the examination. This will be followed by an analysis of what happened and the timing of the events leading up to the seizure of the floppy. This in turn will be followed by an examination of the executable code extracted from the sized floppy and a comparison of that with a running copy of Camouflage.

Examination detail:

In a forensic examination, it is critical to document the examination process including chain of custody, involved parties, image integrity, and analysis steps.

There were some issues outside the scope of this document that caused some delays in this investigation. Proper forensic procedures were maintained throughout this time period to ensure the integrity of this investigation. These procedures include archival copies, md5 hashes of files and procedural documentation.

During this examination, two computers are in use. A Linux-based computer is used for most of the image examination. This computer will be referred to as the “Forensics PC”. There is also a need to run the windows executables and that is done from a computer referred to as the “Test PC”. This Test PC is running Windows 98 and is a non-production computer that is not connected to the lab network. We don’t know what was on the seized floppy so by running in a secure environment, we eliminate the chance having an unknown program cause problems on the lab network. We will also run a copy of the executable program to ensure that evidence is not damaged.

Chain of Custody (how the image was obtained):

There were some questions relating to the image that was obtained from David Keen’s group. This does not place any question on the validity of the evidence, in fact, the file was downloaded a number of times and in all cases, the md5 hash of the file was in fact the same. It seems that the image file was not initially compressed but was saved with a filename indicating that it had been compressed.

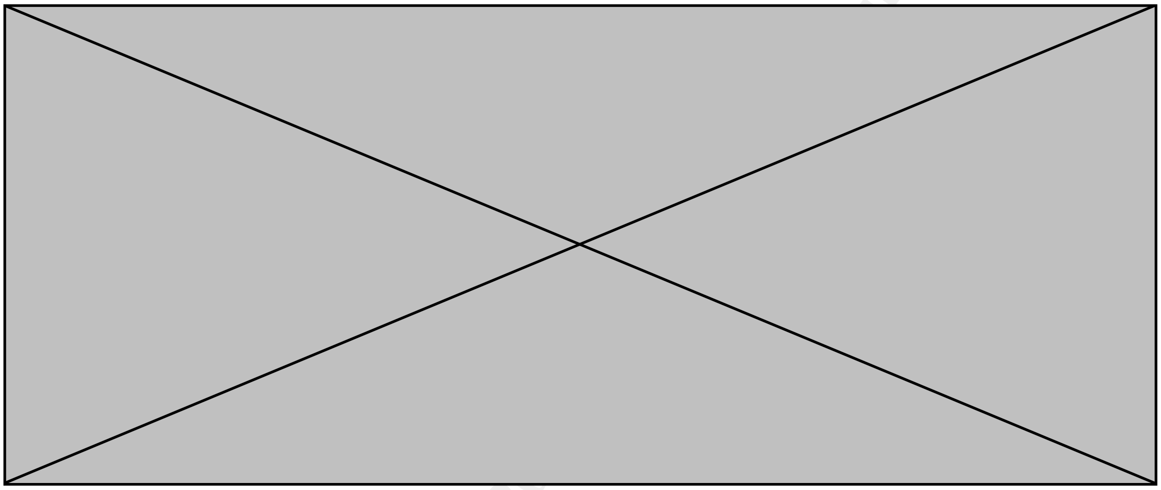
- April 26, 2004 – 16:45 MST – floppy seized from Robert Leszczynski.
- David Keen or other personnel create image of floppy
  - Tag# fl-260404-RLJ-1
  - 3.5 inch TDK floppy disk
  - MD5: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
  - fl-260404-RJL1.img.gz
- June 26, 2004 – 22:12 EST - downloaded v1_5.gz from http://www.giac.org/gcfa/v1_5.gz
- Generate md5sum of v1_5.gz
  - d7641eb4da871d980adbe4d371eda2ad *v1_5.gz
  - file size 1,474,560
  - NOTE: The md5sum of the .gz file is the same as the md5sum of the .img file referenced in David Keen’s notes.
- 22:25 – burn v1_5.gz to cd for safekeeping

    - Tag# fl-260404-RLJ1-org_img
- 22:29 – verify md5sum of image file (v1_5.gz) on CD.
- 22:31 – place CD copy in fireproof safe
- July 21, 2004 19:39 – re-downloaded image to forensic computer using wget on the forensic lab computer to satisfy questions about filename and image integrity from initial download.
    - File size is 502,408 bytes
    - Made copy of file – cp v1_5.gz v1_5.gz.original
- Extracted the image file from v1_5.gz
    - gunzip v1_5.gz
- Checked the md5sum of the image file (v1_5) and it matched the md5sum from the initial download and David Keen's documentation.

Image analysis

Verify image integrity

The md5sum of v1_5 is d7641eb4da871d980adbe4d371eda2ad. That matches initial documentation received from David Keen.



Identify strings in image

We'll use strings on the image to get a quick view of what's on the image. Strings is a program that exists on most UNIX-type operating systems and is available for many others that will extract printable information from a file. Many computer files contain a mix of "human-readable" and binary (looks unintelligible without specific knowledge of what the data means) data. Strings will read through a data file and extract data that can be displayed on the screen. Often this method will allow an examiner to quickly find error messages, copyright information, comments and other "readable data". By running strings on the full image, we can get a quick idea of what the image contains. We can see evidence suggesting a windows environment, we will try to either prove or disprove that hypothesis as we go through the testing. In this case, we're simply wanting to view the data on the screen and get a quick read on what it is so I'm going to pipe the data into less. less is a program that will break scrolling data into "screen-sized chunks". less is simply used to stop the data so that we can view it.

```
strings v1_5 | less
```

This table contains strings and some notes related to those strings. In many cases, the notes were entered throughout the investigation as additional information became available.

AMSHELLDLL	camshell.dll (probably)
INFORM~1DOC	Information_Sensitivity_Policy.doc
INTERN~1DOC	Internal_Lab_Security_Policy.doc*
INTERN~2DOC	Internal_Lab_Security_Policy1.doc* * These filenames may be switched
PASSWO~1DOC	Password_Policy.doc
REMOTE ~1DOC	Remote_Access_Policy.doc
ACCEPT~1DOC	Acceptable_Encryption_Policy.doc
The following did not show up in the mounted image (assume they are from the unallocated space)	
?NDEX HTM	

HTML code including :
NAME=movie
VALUE="ballard.swf"	

SheCamouflageShell	
ShellExt	
CamShell	
BitmapShellMenu	
CamouflageShell	
CamouflageShell	
Shell_Declares	
Shell_Functions	
ShellExt	
modShellRegistry	
kernel32	
lstrcpyA	
shell32.dll	
CreateICA	
C:\WINDOWS\SYSTEM\MSVBVM60.DLL\3	
VBRUN	
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb	
indexMenu	

idCmdFirst	
idCmdLast	
uFlags	
idCmd	
pwReserved	
pszName	
cchMax	
stdole2.tlb	
IctxMenu.tlb	
Information Sensitivity Policy	(seems to be the start of a document related to guidelines for sensitive document handling for Ballard Industries)
Internal Lab Security Policy	(possibly another related document)
Cisco Systems, Inc.	this string is in both documents, don't know why…seems to be in most documents
Password Policy	another document?

Directory listing and structure

By mounting the image as on the forensics computer, can interact with the 'normal' file structure. We have specified a number of options (-o switch) that this image will be mounted Read-Only (ro) to protect it from modification during the examination. We have also used the noexec option to prohibit execution of binaries on the mounted file system. This provides a little extra protection against accidentally executing programs. The loop switch is necessary to mount an image of a drive because it is not an actual block device…the most common example of a block device is a hard drive.

```
mount –o ro,loop,noexec v1_5 /mnt/floppy
```

Once the image is mounted, we can interact with the file structure of the drive. First, we'll get some directory usage information from the image. This tells us that the disk is 46% used meaning that there may be 54% of the disk with data that the operator may think was deleted but that may have evidentiary value.

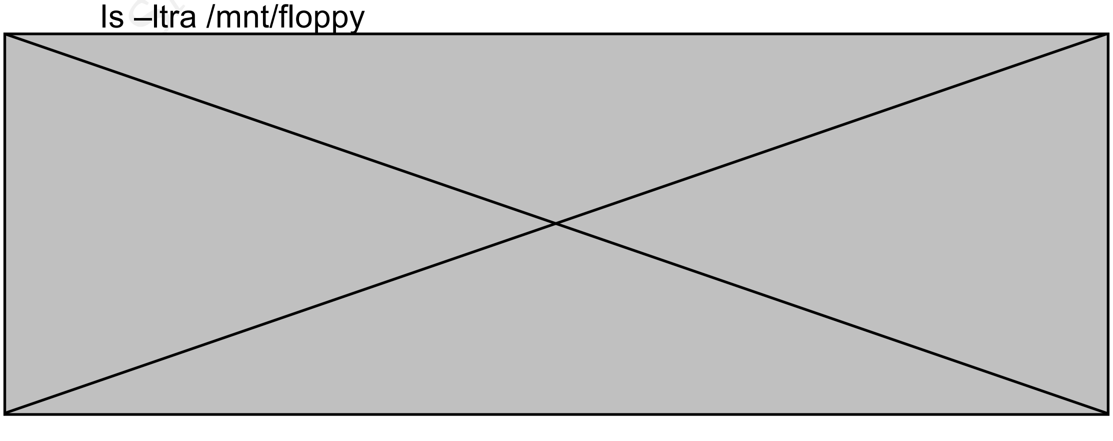
```
df /mnt/floppy
```

Next, we'll get a directory of the mounted image from both the mounted image and the unmounted image. The directory from the mounted image is retrieved using the ls command. The directory from the unmounted image is retrieved using fls gives information about deleted files.

ls (lower case L and lower case S) is a standard command on most UNIX-like systems. Since this forensics computer has this built in, ls is available for getting a 'normal' directory. This is comparable to the dir command on a DOS or Windows-based computer. The command-line switches we will use are l (to give a long display – date, size, permissions, etc.), t (sort by time), r (reverse the sort) and a (all – do not hide any entries). In most forensic analysis, time plays an important role so I've chosen to sort the directory based on time to make things match up with the time-line a little more intuitively. One additional switch that we could have used here is the –R switch to recurs through the directory structure. There are directories other than the current directory (.) and the prior directory (..). Directory entries would be noticeable because of the d in the first column of a particular line in the listing.

The –l option referenced above also includes the owner of the file. In this case, we see root listed. In this case, that is the identity of the auditing account, and not information retrieved from the file because fat12 doesn't support the concept of ownership.

ls –ltra /mnt/floppy

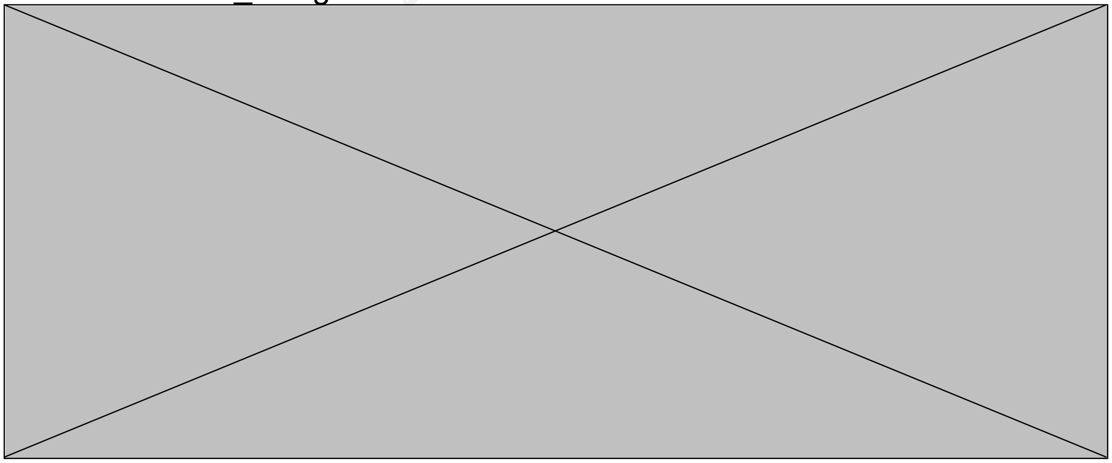


NOTE: The following step was performed after lazarus was run. For purposes of clarity and comparison, it is included here.

The fls program is part of "The Sleuth Kit"[2]. TSK as it's often referred to is a collection of tools that can be used to analyze a file system. TSK and related programs are favorites of forensic investigators. Utilities from TSK will be used throughout this examination. The major thing that fls gives us here is the full filename as well as the "DOS filename" (truncated to conform to legacy DOS rules). In some cases, this may be an advantage as we go through the analysis. By default, fls will also display deleted directories. In a forensic investigation, deleted information is often the most interesting. We only need two switches for this analysis, they are the –f switch which specifies the file system (more details on that on the next page) and the –r switch to recurs through all directories contained within the current one. In this case, the –r switch didn't make any difference but it's still a good thing to use that switch to ensure that we're seeing everything.

In viewing this information, the r at the beginning of each entry indicates that these are files. If they were directories, they would be noted with a d. The number followed by a colon (:) is the inode number. This utility has quite a number of additional switches. Some of those will be used a little later when we build a timeline of file activity.

fls –f fat12 –r v1_5.img



[2] http://sleuthkit.sourceforge.net

Get file system stats of image

At this point, we don't know what the file system is. Since it's a floppy and it came from a user and there are indications that it's a windows-based system from the collected strings, we strongly suspect it would be fat12. The forensic computer determined this when it as a vfat partition when the imiage was mounted. We can see this by running the mount command and listing all the mounted partitions and the information about them. Here we see the full path of the image, the mount point and the options used to perform the mount.

```
mount

/forensics/unixforensics/GCFApart1/rlj/images/v1_5.img on /mnt/floppy
type vfat (ro,noexec,loop=/dev/loop0)
```

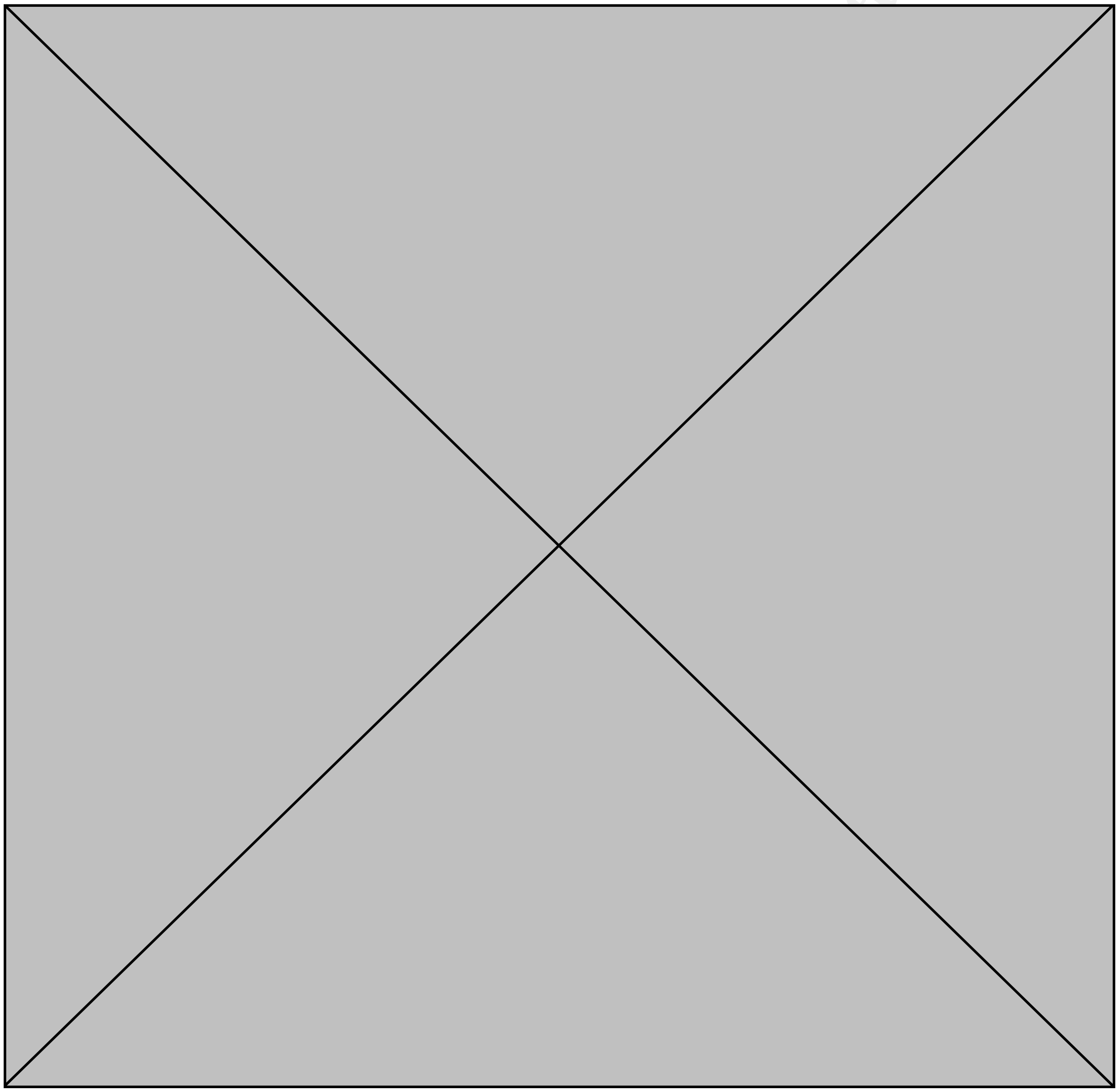
Another tool that will help us identify the file system on the floppy is file. File is a program that can identify computer files by their specific characteristics. Since hard drive formats are very rigidly structured, this will give us very good evidence of the type of file system that we are working with.

```
file  v1_5.img
v1_5.img: x86 boot sector, system mkdosfs, FAT (12 bit)
```

It seems quite clear that this is in fact a floppy with a fat12 format. We will pass this information to the other utilities that we use to examine the floppy image.

The tool fsstat is another part of TSK. This utility will read through the image and provide us with status information from the image. We will use the –f switch to pass the file system type to the program. If this information were incorrect, should get an error indicating that. In this case, fsstat provides additional support for the fact that this is in fact a fat12 formatted disk. Note that the volume label is RJL – these are the initials of the person the disk was seized from.

```
Fsstat –f fat12 v1_5
```



Unallocated data

The next step in examining the images is to extract unallocated data from the image to a separate file for analysis. Often the most interesting forensic data comes from files that somebody thought they deleted. In most cases, when data is deleted from a disk, it's not really deleted; it's just flagged as being unallocated and available for use. In this case, we'll find a fragment of a windows dll that will prove to be the key to cracking the encryption.

We'll use dls (part of TSK) to extract the unallocated data from the floppy. Unallocated data could be space on the disk (or image) that has never been used or it could be space that is flagged as unallocated so that it can be used. Our image is named v1_5 and we'll create a file with the unallocated space named v1_5.dls.

- dls –f fat12 v1_5 > v1_5.dls
- size of extracted data is 797,696 bytes. This matches the 780 1K blocks that the basic stats in prior checking reported.
- md5sum of the unallocated space on the image
  fea7fa324510a93ea893da11ee57ba53 v1_5.dls

We can now analyze the unallocated part of the image using Lazarus, a part of The Coroner's Toolkit[3]. Lazarus is a forensic tool that will analyze raw data and attempt to identify disk blocks. We're going to give Lazarus the deleted file space and see if it can come up with anything interesting. Since we really don't know exactly what we're looking for at this point, this may help give some more clues. Actually, we have some pretty good ideas but we need more details. Since this is a very small image Lazarus will finish this very quickly. If this were a full disk image and we were in a hurry, we'd need to pick another method as Lazarus can be horribly slow.

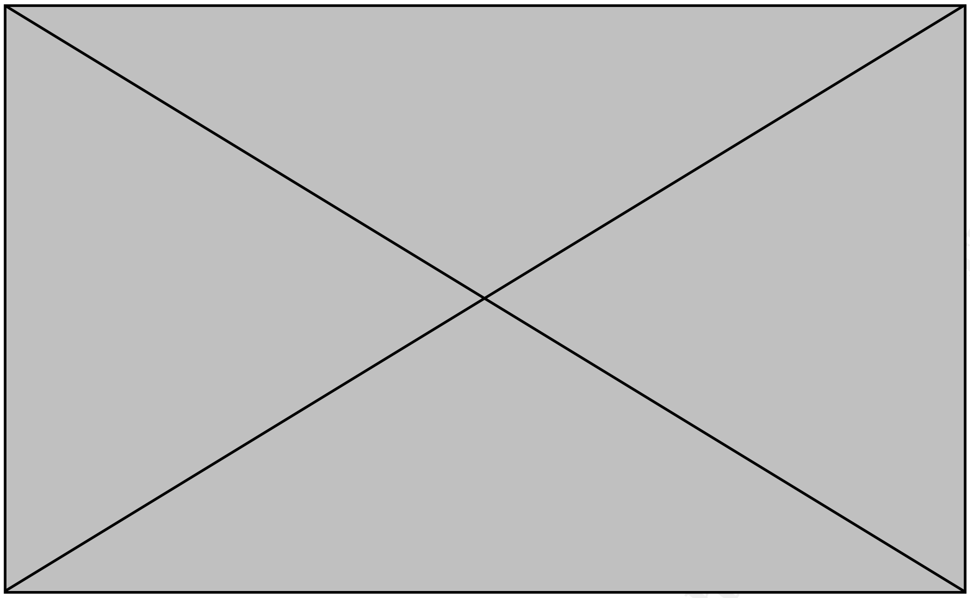
This will be a two-part process. First we'll use Lazarus to analyze the data and create html-based files. We can then use Mozilla (a web browser on the Forensics PC) to view the html pages that Lazarus generates.

```
lazarus –h v1_5.dls
mozilla v1_5.dls.frame.html
```

In the initial Lazarus screen, we can see the data that was extracted. The top section of the web page is the legend. In comparing the data in the window with the legend, it looks like we have an HTML page (yellow H), some binary data (…..) an archive (Aaaaa) and then a sound file (!!!!!!!!!!). These are just guesses based on file characteristics. We'll need to look at them individually for further

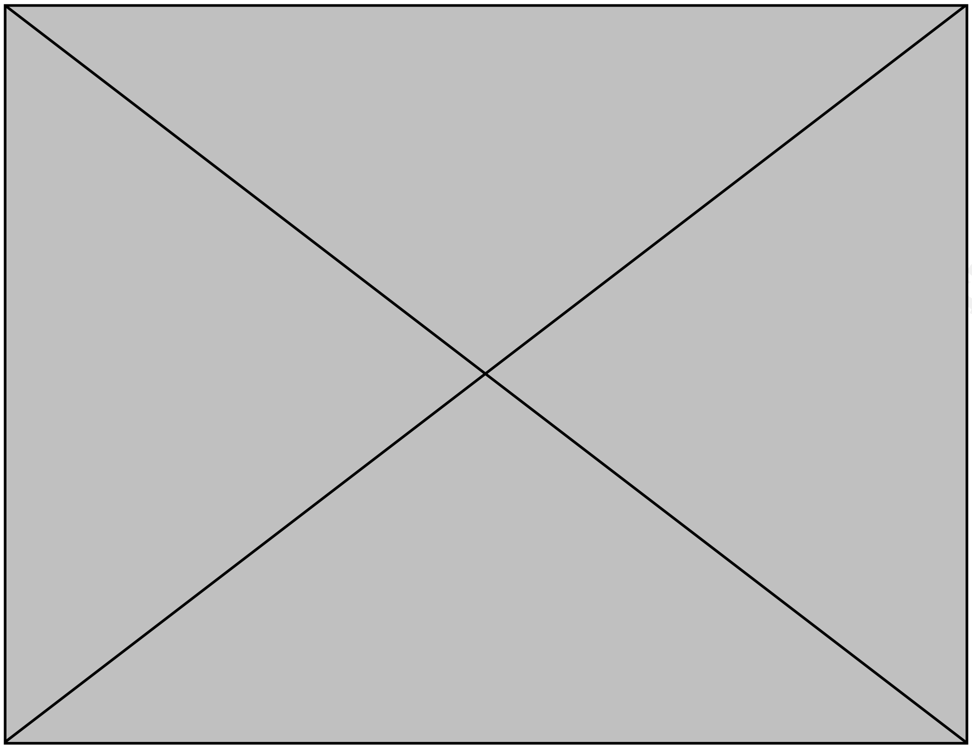
[3] http://www.porcupine.org/forensics/tct.html

identification.

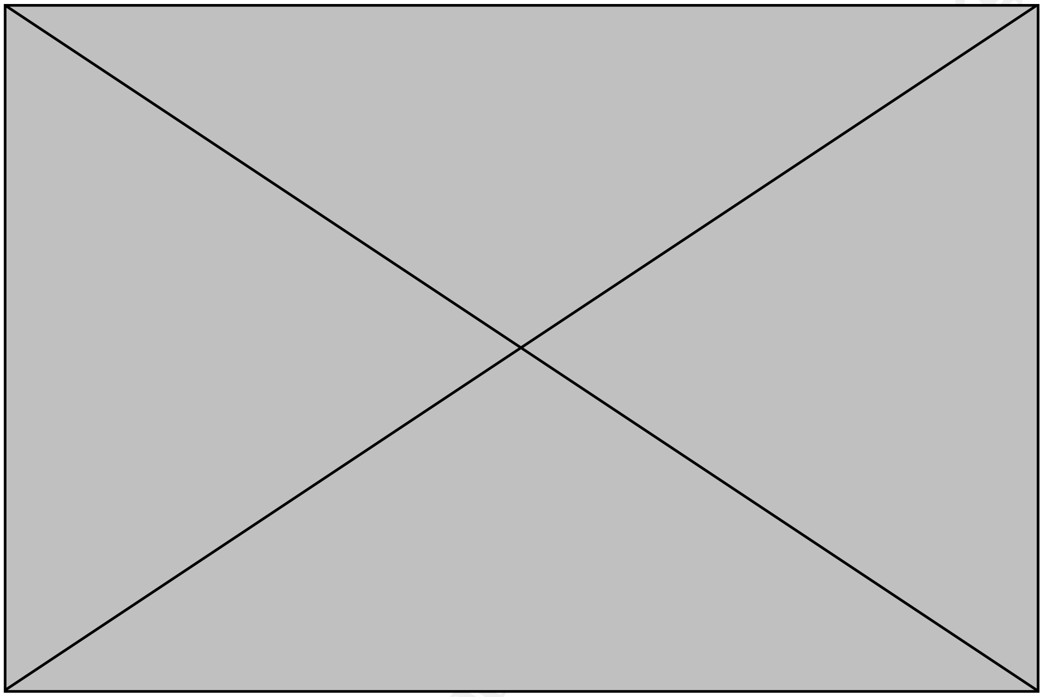


We’ll look at these blocks in the order they are displayed. That’s also the order they are on the disk.

The first one is the html block. This is a copy of the html page that was in the unallocated space on the floppy. Notice the id=“ballard” and VALUE=”ballard.swf”. This floppy came from Ballard Industries – at this point, we don’t know what if any significance this file may have.



The next block is marked as an archive file by Lazarus. If we look at the data on the 2nd line, we see a reference to a web site – http://www.camouflage.freeserve.co.uk. This matches some of the information we got earlier from our string search on the drive. We saw a Camouflage directory reference on a C: drive; there is also a CamouflageShell and a few other similar references. The fls directory also showed a camshell.dll file.

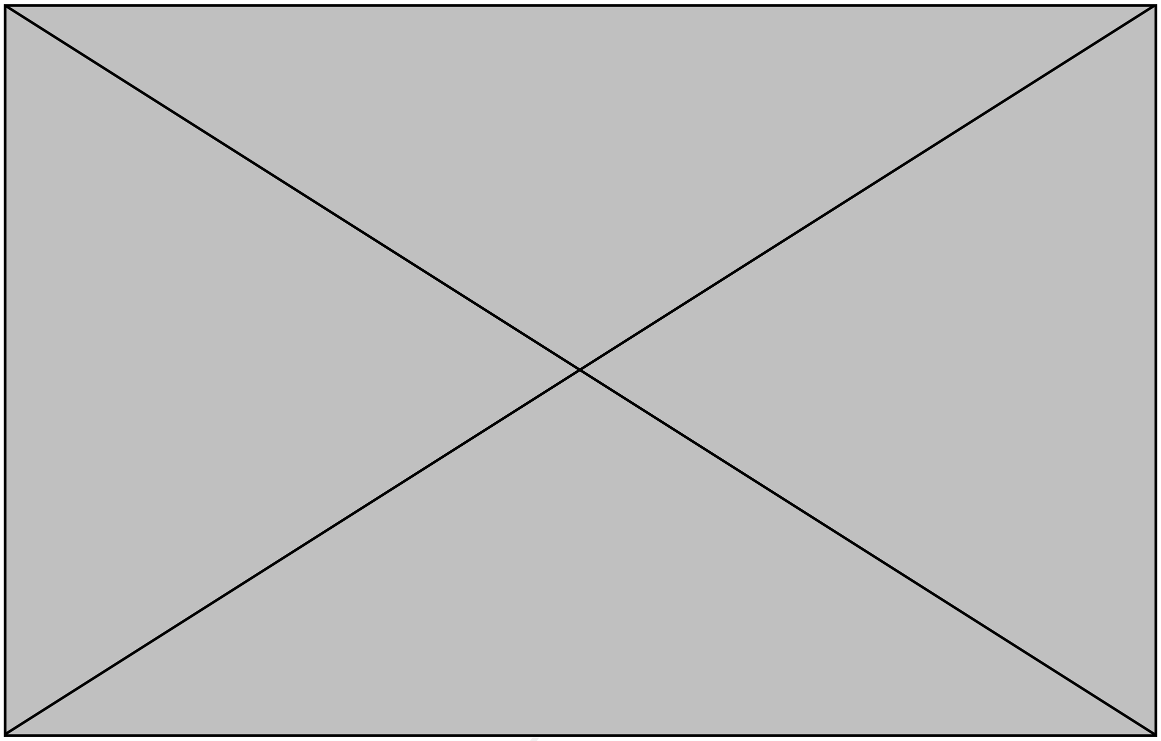


The following table is some of the interesting data collected from this particular block of deleted space.

Typelib	
Shellext	
MSVBVM60.DLL	Visual Basic file
_adj_fptan, _vbaVarMove, _vbaFreeVar, etc	Looks like programming code – variables or functions
http://www.camouflage.freeservice.co.uk	Doesn't exit, now http://camouflage.unfiction.com/
Twisted Pear Productions	
Keeps files containing sensitive information safe from prying eyes.	
Copyright 2000, 2001	
Product name Camouflage 4	
Version 1.01.001	

Camshell.dll
Stringfile info 040904B0

The next block is marked as a sound file. We can see some text in this file including CamouflageShell which one of the interesting strings we noticed in the initial string search of the full image.



Create a timeline

The next task is to create a timeline of the image. Since this is a windows operating system, this will show us when files have been modified, accessed or created. This will aid in correlating the times things happened on the disk with the time of the seizure, times that certain parties (namely Robert J. Leszczynski – but we must be careful not to limit the investigation prematurely) had access to the R&D lab, Robert's briefcase, etc.

The fls command will extract timeline information from the file system information. The ils command will extract timeline information from the allocated and unallocated space with some additional information. Then all of that data will be merged into a single file (v1_5.mac) and finally mactime will be used to pull the information into one nice report as seen below.