GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

Interested in learning more?

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

"It is an important and popular fact that
things are not always what they seem."[1]

GIAC GCFA
Practical Assignment

Version 1.5 (April 30, 2004)
SANS 2004, Orlando, Florida

Daniel Wesemann
July 30, 2004

---

[1] Douglas Adams, The Hitchhiker's Guide to the Galaxy, Chapter 23

"All these things will become clear to you", said the old man gently.
"At least", he added with slight doubt in his voice, "clearer than they are at the moment".
---Douglas Adams*,*
Hitchhiker's Guide to the Galaxy

Abstract

This document has been prepared as an assignment to qualify as a GIAC Certified Forensic Analyst (GCFA). The paper has two parts, following the structure of the two tasks set by the GIAC practical assignment for Forensic Analysis, Version 1.5.

Part one is an analysis of an unknown floppy image, showing all steps of the forensic process, and also containing a brief excursion into the neighbouring field of reverse engineering of software and encryption schemes to solve a tricky forensic riddle.

Part two is an analysis of an unknown hard drive, acquired through an online auction for less than one dollar. The vendor formatted the hard drive before handing it on, which made the analysis harder, but far from impossible. Read on to find out just how much data is hidden on an apparently empty disk drive.

In the course of this assignment, two short scripts were created that can help the forensic analyst with extracting Microsoft Executables and Internet Explorer Browser History from a formatted drive. Both tools will be made available to the forensic community.

Table of Contents

Part 1 – Analyze an Unknown Image

Management Summary

Ballard Inc have contracted our forensic investigation services in order to analyze a floppy disk seized from a suspect on April 26, 2004. We applied our experience and solid forensic methodology to the task, and in the course of our investigation were able to prove that

1. The seized disk indeed contains confidential information of Ballard Inc, including an excerpt from a customer address database and proprietary drawings of fuel cell technology
2. Encryption technology not authorized for use in Ballard Inc. has been employed to hide this confidential information inside of benign looking public documents
3. One of the hidden files contains an offer to provide proprietary information for USD 5 million, signed with the suspect's full name
4. The method used to hide information, as well as the content of the message, suggests that this is not the first contact between the suspect and the unknown intended recipient of the material. While this floppy was seized by the security guards before it could be carried off company premises, we have to assume that earlier attempts were successful.

In view of the above findings, we strongly urge Ballard Inc. to conduct a full scale forensic investigation of the workstations and data servers used by the suspect. This will help to potentially reveal further evidence and also to better assess the full extent of the damage.

Image Source and Acquisition

A floppy disk was seized from the suspect on April 26, 2004, around 4:45 pm MST, and subsequently handed over to the company security administrator, David Keen. We were provided with an image copy of the floppy disk on July 3, 2004, at 08:20 am MST, and immediately started with the analysis. The floppy image came accompanied with the following chain of custody information:

> Tag Number: fl-260404-RJL1
> Description: 3.5 inch TDK floppy disk
> MD5 Hash: d7641eb4da871d980adbe4d371eda2ad
> Image Name: fl-260404-RJL1.img.gz

As a first step in the process, we created an additional copy of the image for analysis purposes and then verified that the MD5 hashes of the original, the image and the analysis copy were indeed the same. MD5 hashes are a cryptographic method to verify that files are identical – the checksum function of MD5 has been designed in such a way that even small deviations in the file content will result in a completely different MD5 hash. It is generally accepted practice in the computer forensic community that a copy of a file is a true image of the original if the two MD5 sums are the same.

Original Floppy MD5:   d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1
Image provided by Mr. Keen: d7641eb4da871d980adbe4d371eda2ad v1_5
Copy for analysis purposes: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.cpy
Floppy for analysis purposes: d7641eb4da871d980adbe4d371eda2ad /dev/fd0

Examination Process

We then proceeded to identify the type of file system on the image in order to chose
the appropriate analysis tools for the next steps. Assuming that the floppy had been
used on a Windows or Linux system, we ran the fsstat tool with the parameter that
is most common for floppies created on these platforms. The output and associated
explanations are shown in the appendix.

The most evident facts uncovered by this first analysis step are:

1. The floppy has likely been initialized (formatted) on an Unix/Linux system,
   because the software creating the "mkdosfs" OEM tag normally runs on Unix
2. The total resulting size of 2781 sectors is 8 sectors smaller than what is
   common for this floppy format, indicating that the formatting process could
   have been tampered with or that the floppy contains bad (damaged) sectors.
3. The disk contains six visible files which occupy a contiguous section on the
   floppy. This is an indication that the files were sequentially copied onto the
   disk.

In order to get an overview on the amount of potentially deleted or hidden data on
the floppy, we created the graphical map of the floppy contents shown on the
following page. The map shows the entire floppy, with each character representing a
data block.

Content shown in green represents allocated space that either belongs to the six
files mentioned earlier, or that has been occupied by the operating system to hold
auxiliary data. Green blocks marked with a "#" hash sign actually hold data, whereas
sections marked with a "\" backslash are empty but still belonging to a file. Content
shown as black dots represents unused space, which either means that these
sections of the floppy have never been used to hold data, or that the corresponding
blocks have been deliberately and irrevocably overwritten. Sections shown in red
and marked with a "D" are the most interesting, since these blocks appear to contain
or have contained data, but are marked as "unused" in the master index of the floppy
allocation list.

From this graphical map, it was immediately apparent that only about the first half of
the floppy actually contained data, and that there were some sections with hidden or
deleted data that would warrant further examination.

```
          0....5....0....5....0....5....0....5....0....5....0....5....0....5....0....5....
00000000 #####\\\\\####\\\\\\##\\\\\\\\\\\\DD......DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD.
0000a000 .........DDDDDDD.DDDD....####\###############################################
00014000 ######\\\\\\#\\\\\\##########\###############################\\\\\\#\
0001e000 \\\\\##########\###############################\\\\\\#\\\\\\\###########
00028000 #\###############################################\\\\\\#\\\\\\###########
00032000 ################################################################################
0003c000 ################################################################################
00046000 ################################################################################
00050000 ################################################################################
0005a000 ################################################################################
00064000 ################################################################################
0006e000 #######################################\########################################\\
00078000 \\\\\#\\\\\\#################################################################
00082000 ################################################################################
0008c000 ################################################################################
00096000 ################################################################################
000a0000 #####################################################\###############
000aa000 #\\#\\\\\\\#\\\\\\######..........................................................
000b4000 ................................................................................
000be000 ................................................................................
000c8000 ................................................................................
000d2000 ................................................................................
000dc000 ................................................................................
000e6000 ................................................................................
000f0000 ................................................................................
000fa000 ................................................................................
00104000 ................................................................................
0010e000 ................................................................................
00118000 ................................................................................
00122000 ................................................................................
0012c000 ................................................................................
00136000 ................................................................................
00140000 ................................................................................
0014a000 ................................................................................
00154000 ................................................................................
0015e000 ................................................................................
00168000
```

Before trying to reveal any hidden secrets, though, we took a brief look at the contents of the six files obviously present on the floppy. Listing the directory of the floppy showed six Word documents, apparently information security policy documents, one of which was stored in two versions. We briefly read through the content of these files and did not encounter anything out of the ordinary, letting us assume that these documents were indeed copies of the security policies used at Ballard Inc. There were two notable things that stood out, though. For one, the text content of the two copies of the Lab_Security policy appeared to be identical, but the file sizes were different. For another, the Password and Remote_Access policies both only contained three pages of text, but still occupied as much space on the floppy as a document ten times the size would need. Both discoveries suggested that the files could contain hidden information which was not being shown when the files were opened with Word or OpenOffice.

Image Details

Using forensic tools like `fls` and `ils` (details shown in appendix), we were able to obtain the following detail information about the files on the floppy.

Type	Inode	Long File Name	Short File Name	Modification Date and Time		Access Date			Creation Date and Time			Size	UID	GID
r/r	3:	RJL	(Volume Label Entry)	2004.04.25	10:53:40 (MST)	2004.04.25	00:00:00	(MST)	2004.04.25	10:53:40	(MST)	0	0	0
r/r *	5:	CamShell.dll	(_AMSHELL.DLL)	2001.02.03	19:44:16 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:18	(MST)	36864	0	0
r/r	9:	Information_Sensitivity_Policy.doc	(INFORM~1.DOC)	2004.04.23	14:11:10 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:20	(MST)	42496	0	0
r/r	13:	Internal_Lab_Security_Policy1.doc	(INTERN~1.DOC)	2004.04.22	16:31:06 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:22	(MST)	32256	0	0
r/r	17:	Internal_Lab_Security_Policy.doc	(INTERN~2.DOC)	2004.04.22	16:31:06 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:24	(MST)	33423	0	0
r/r	20:	Password_Policy.doc	(PASSWO~1.DOC)	2004.04.23	11:55:26 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:26	(MST)	307935	0	0
r/r	23:	Remote_Access_Policy.doc	(REMOTE~1.DOC)	2004.04.23	11:54:32 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:36	(MST)	215895	0	0
r/r	27:	Acceptable_Encryption_Policy.doc	(ACCEPT~1.DOC)	2004.04.23	14:10:50 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:46:44	(MST)	22528	0	0
r/r *	28:		_ndex.htm	2004.04.23	10:53:56 (MST)	2004.04.26	00:00:00	(MST)	2004.04.26	09:47:36	(MST)	727	0	0

Shown are two deleted files (CamShell.dll and _ndex.htm) which were not visible in the normal directory listing. The modification and creation time stamps suggest that all files had been prepared (modified) elsewhere and were then copied (created) to the floppy on April 26, 2004, around 9:45 am MST, seven hours before the floppy was seized by the Ballard security guard. The time stamp of the volume label indicates that the floppy has been initialized (formatted) one day earlier, on April 25, 2004, at 10:53 am.

Retrieving deleted files

Using the forensic browser `autopsy`, we were able to retrieve the (partial) contents of the two deleted files. This is possible because a computer does not actually delete a file when told to do so – it simply flags the corresponding space as "ready for re-use", allowing the original data to be overwritten should the need arise. As long as this doesn't happen, the original contents remain accessible.

Filename	CamShell.DLL	_ndex.htm (likely index.htm)
Size	36864	727
Sectors	33-104	33-34
File Type	HTML document	HTML document
Modified	Sat Feb 3 19:44:16 2001	Fri Apr 23 10:53:56 2004
Accessed	Mon Apr 26 00:00:00 2004	Mon Apr 26 00:00:00 2004
Created	Mon Apr 26 09:46:18 2004	Mon Apr 26 09:47:36 2004

Both deleted files are shown to use sectors 33 and 34, and the content type of "HTML" is not what would be expected for a binary DLL. This means that most likely the first two blocks of CamShell.DLL have unfortunately been overwritten when _ndex.htm was copied to the floppy as well. Thus, while we can recover the entire contents of _ndex.htm, recovery of CamShell.DLL is only partial.

_ndex.htm

The content of the recovered file _ndex.htm is shown below. The original file name most likely was "index.htm", the common file name used for an entry page on a web server. This, together with the contents of the file, makes us assume that _ndex.htm is indeed a copy of the start page of Ballard's public Internet offering. As such, the presence of this file on the floppy is nothing out of the ordinary.

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

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

CamShell.DLL

CamShell.DLL is a program file that would need to be translated from machine language back to human readable format in order to understand what the program was used for. As a first step, we extracted all the text strings from the recovered file, hoping to find clues about the purpose or origin of the file. The actual extraction procedure is shown in the appendix, the most interesting text strings (keywords) found in the file are:

```
CamouflageShell
C:\My Documents\VB Programs\Camouflage\Shell\IctxMenu.tlb
Camouflage.exe
Software\Camouflage\Settings
http://www.camouflage.freeserve.co.uk
Keeps files containing sensitive information safe from prying eyes.
Copyright © 2000-2001 by Twisted Pear Productions
FileVersion 1.01.0001
```

This definitely looks like we are onto something. Unfortunately, the URL listed in the file is no longer available – but search engines like Google have a long memory. Using "camouflage twisted pear" as search term in Google, we were able to locate the new home at http://camouflage.unfiction.com. The page provides a "download section", offering a self-extracting Camouflage v1.2.1 for download. According to the associated documentation, Camouflage can be used to hide one or several files within another "host" file. During this process, the host file increases in size, but retains its time stamp and can also still be opened with the original program without revealing anything of its new contents. To protect the hidden information even further, Camouflage also allows the user to specify a password.

In order to establish whether the CamShell.DLL found on the evidence floppy was indeed part of this Camouflage tool, we downloaded the software and installed it on a Lab system, resulting in the files shown below.

```
Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: D86C-8322

 Verzeichnis von C:\Programme\Camouflage

04.07.2004  09:32    <DIR>          .
04.07.2004  09:32    <DIR>          ..
29.03.2001  22:13         217'088 Camouflage.exe
03.02.2001  19:44          36'864 CamShell.dll
28.03.2001  19:50          11'649 Readme.txt
04.07.2004  09:33          19'701 Uninst.isu
              4 Datei(en)        285'302 Bytes
              2 Verzeichnis(se), 11'799'150'592 Bytes frei
```

A file CamShell.dll bearing the same time stamp as the file retrieved as evidence was immediately apparent. To be certain, we went about to compare the contents of the two files. Since the first two blocks of our evidence file were corrupt (overwritten by the HTML file), we extracted from both the evidence and the installation DLL all content save the first two data blocks (details shown in appendix).

Then, we used MD5 to verify if the files were the same. And, yes…

```
aaf222265674efd802361f560f305a74    fl_clean_camshell.dll   [file from evidence]
aaf222265674efd802361f560f305a74    clean_camshell.dll      [installed original]
```

This indicates that Camouflage v1.2.1 from Twisted Pear Productions may have been used by the suspect to "hide" information on his computer and possibly also on the floppy under investigation.

Retrieving hidden content

Our earlier suspicion that the policy documents on the floppy might be more than what meets eye was thus reinforced to the point where we actually ran the Camouflage tool against the contents of a floppy recreated from the image.
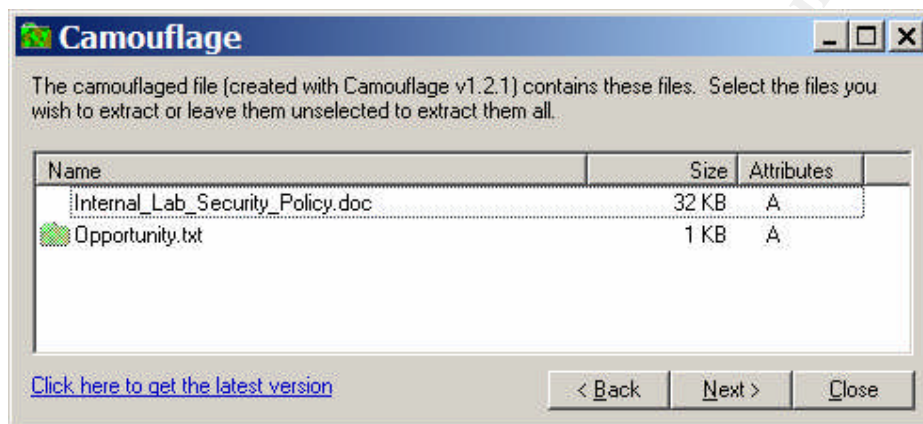

And indeed, one of the files on the floppy was showing "camouflaged" information that could be retrieved without a password. The hidden file, Opportunity.txt, was extracted and is shown below.
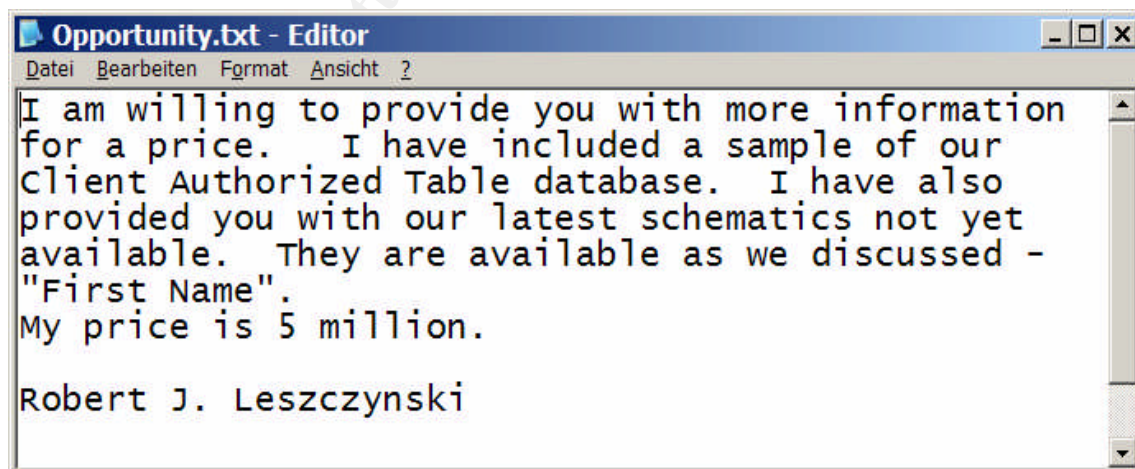
This finding establishes that an offer to sell trade secrets bearing the signature of the suspect has been hidden in a policy document by using Camouflage v1.2.1 as encryption tool. While the suspect could have been framed by an other employee,

the total evidence strongly suggests that the tool indeed has been used by the suspect to deliberately hide information.

The hidden content of the other files was not immediately accessible, because a password had been used to further protect the camouflaged content. Even though the hint in the Opportunity.txt file – "First Name" – was an obvious suggestion to use the first name as password, we were at first unable to reveal the hidden contents of the other files. We began with a meticulous string search of the entire floppy image, thinking that "first name" was referring to the name of a person, but nothing. The names of the suspect did not work as password either.

Camouflage

⚠ Either the file requires a password or it is not camouflaged.

OK
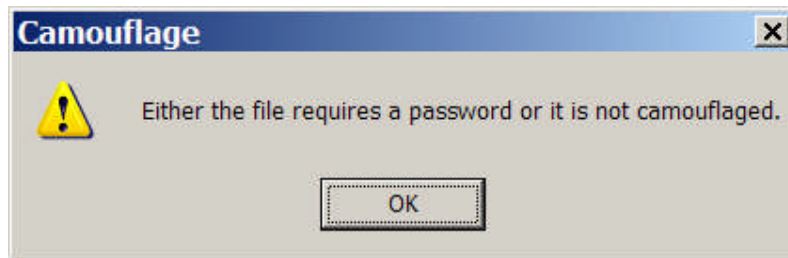
Assuming that the contents of the remaining files were of utmost importance to the case, but unable to get any further, we finally started to reverse engineer the encryption algorithm used by the tool. Details of this procedure are listed in the appendix – but basically, we used the Camouflage tool to encrypt known contents with a known password, and then compared the resulting cipher texts in an attempt to reveal some of the inner workings of the tool. During this process, we uncovered two elements of utmost importance:

Every file that contains information hidden with Camouflage has a distinct "end of file marker" of 0x74,0xA4,0x54,0x10,0x22,0x97 near the end of the last data block. This indicator can be used to quickly locate files that might contain hidden information. The appendix contains a Perl script that can be used with this pattern in order to rapidly decide whether a file warrants further investigation. This information will be very helpful during the ensuing investigation of file servers and network storage at Ballard Inc.

Secondly, while trying out Camouflage with passwords of various length, we noticed that a data structure near the end of the encrypted file seems to vary in sync with the length of the password used. The appendix shows how this discovery was used to reverse engineer the password obfuscation mechanism of Camouflage, and to subsequently retrieve the passwords used to protect the hidden contents within Password_Policy.doc and Remote_Access_Policy.doc. The corresponding passwords are "Password" and "Remote", which finally sheds some light onto the "First Name" hint found in the earlier file: It did not refer to a person's first name, but rather to the first component of the filename containing the hidden data.

With this information, we were finally able to extract the contents from the floppy to a forensic folder on our lab system. Hidden within Password_Policy.doc were three pictures, and Remote_Access_Policy.doc had been used to camouflage a database.
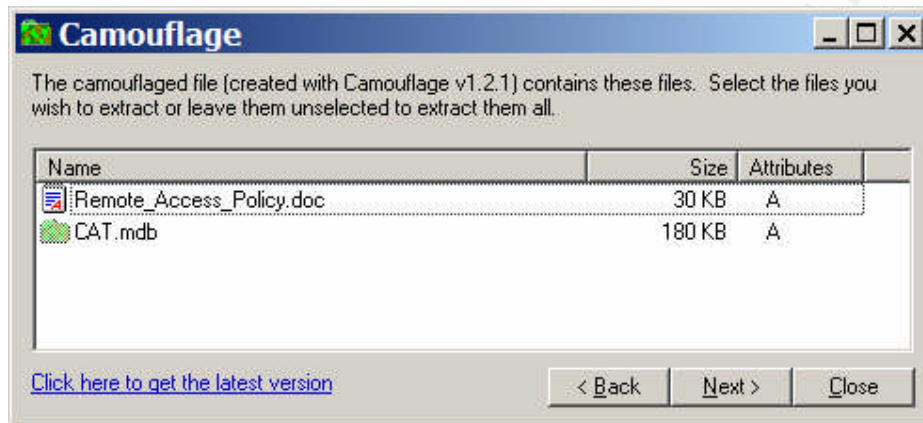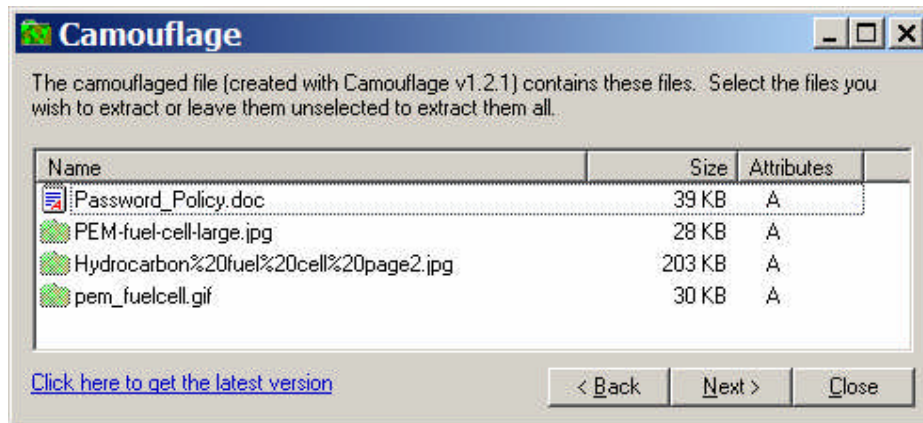
The file names and associated MD5 checksums shown below will allow at any time to tie the extracted files back to the evidence image.

```
c3a869ff6b71c7be3eb06b6635c864b1 *CAT.mdb
9da5d4c42fdf7a979ef5f09d33c0a444 *Hydrocarbon%20fuel%20cell%20page2.jpg
3ebd8382a19c88c1d276645035e97ce9 *Opportunity.txt
5e39dcc44acccdca7bba0c15c6901c43 *PEM-fuel-cell-large.jpg
864e397c2f38ccfb778f348817f98b91 *pem_fuelcell.gif

C:\Data\Forensic>dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: D86C-8322

 Verzeichnis von C:\Data\Forensic

04.07.2004  18:17    <DIR>          .
04.07.2004  18:17    <DIR>          ..
23.04.2004  19:21           184'320 CAT.mdb
23.04.2004  18:21           208'127 Hydrocarbon%20fuel%20cell%20page2.jpg
23.04.2004  22:03               312 Opportunity.txt
23.04.2004  18:23            28'167 PEM-fuel-cell-large.jpg
23.04.2004  18:15            30'264 pem_fuelcell.gif
               5 Datei(en)        451'190 Bytes
               2 Verzeichnis(se), 29'144'784'896 Bytes frei

C:\Data\Forensic
```
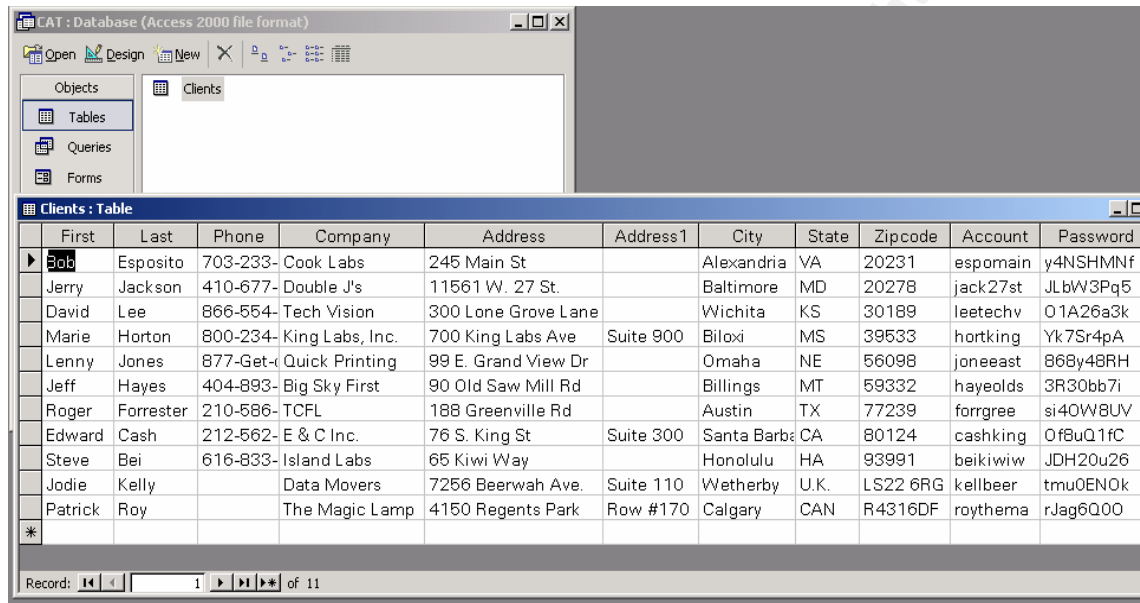
Conclusion

The collected evidence shows clearly that the suspect was trying to smuggle three images (.jpg and .gif) as well as one database file (.mdb) off the premises of Ballard Inc. by hiding the data on a floppy containing benign looking security policy files. The images contain drawings of technical nature, whereas Hydrocarbon*page2.jpg appears to be a single page excerpt of a research paper on the subject of fuel cells. CAT.mdb is an Microsoft Access 2000 database containing contact information for clients of Ballard Inc, as shown on the screenshot below. All retrieved files have been provided to Ballard Inc for further examination.

From the modification dates of the retrieved files, it appears as if they were last modified late in the evening on Friday, April 23. Together with the timeline of the floppy contents, this could mean that the process of hiding the information was already performed on Friday evening, and that on Monday morning, the suspect only copied the content onto the floppy. This is of importance because we assume that residual data or even a copy of the evidence itself could also be found on the hard drive of the suspect's PC.

Since more camouflaged data might be hidden on the systems of Ballard, Inc, we strongly urge the company to extend our contract to allow forensic investigation of Ballard workstations and file servers. Minimally, the system administrators of Ballard, Inc, should use the script provided in the Appendix of this report to search for and reveal camouflaged content.

Legal implications of uncovered activities in Switzerland

Using cryptographic software to hide information is not illegal in Switzerland. But most companies have corporate policies covering the handling of proprietary information, and using a non-standard encryption tool obtained from the Internet in order to hide confidential data is most certainly outlawed by all company policies we know. Attempting to carry confidential information off premises for personal financial benefit is a gross violation of company policies and also of Swiss Federal employment contract law, as shown below.

If an employee provides proprietary and confidential information of his employer to a third party, he/she is in violation of OR Art. 321a II/4 of the Swiss Obligations Code. This section of the code states (translated to english)

An employee must not make use of or provide to third parties any proprietary information like trade or manufacturing secrets of which he becomes aware in the course of his work. This rule remains in effect even after termination of an employment contract, if so needed to protect the interests of the employer.

Also, OR Art. 321e VI/2 is explicit on liabilities

An employee is liable for any deliberate or negligent damage to the employer.

If the information sold to a third party ends up in a foreign country, and the information is deemed to be vital or technologically highly advanced, the activity might count as industrial espionage, covered in StGB Art. 273 (Criminal Code)

Who obtains manufacturing or trade secrets in order to provide them to a foreign office, organization or corporation will be punished with prison not under one year and in severe cases up to twenty years..

Given the evidence in the case at hand, most Swiss corporations would attempt to settle with the suspect and the recipient of the information outside of a court of law. If the case were taken to a court, additional evidence or a confession of the suspect would likely be necessary to reach conviction. Without also investigating the personal workstation of the suspect and also encountering corroborating evidence there, the case would be thin -- the suspect could always claim that he had been framed by another employee and never was aware that the floppy had been placed into his briefcase. Still, if convicted in a Swiss court, the suspect would face a prison term of up to two years and a fine.

Appendix to Part #1

Most of the analysis was performed on a Lab system using Windows XP as base operating system and running copies of SuSE Linux 8.1 and Windows XP within virtual "VMWare" systems. Not knowing what to expect, we performed the very initial analysis on a separate system, booting it from a Helix 1.4 CD. Sound forensic practice asks for a verification of the integrity of the analysis environment before starting with the analysis. Given the complexity of verifying a multiboot VMWare setup from a forensic viewpoint, this step was not performed, but all checksums obtained on the analysis workstation were verified on the separate Helix system.

Method used to identify file system on floppy

```
daniel@cholla$ fsstat -ffat fl-260404-RJL1.cpy

FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT
```

[Comment: mkdosfs is a tool commonly found on Unix. This floppy has likely *not* been initialized/formatted on a Windows system]

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

Sectors before file system: 0

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

META-DATA INFORMATION
--------------------------------------------
Range: 2 - 45426
Root Directory: 2

CONTENT-DATA INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 512
Total Cluster Range: 2 - 2840

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

[Comment: Floppy apparently contains six files]

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

Method used to create the graphical map

The script was custom made to the task, and contains hard coded values as returned by fsstat. Don't use without modification on other images.

```perl
#!/usr/bin/perl

sysopen IMG,$ARGV[0],"RO" || die "hmm";
binmode (IMG);
$offset=0;$size=512;
$count=sysread IMG,$data,$size;
print "
0....5....0....5....0....5....0....5....0....5....0....5....0....5....0....5....\n";
printf "%08x ",$offset;
while ($count>0) {
        #$data=~tr/[a-zA-Z0-9]/x/c;
        my $first=substr $data,0,1;
        my $subst=$first x 512;
        if ( ($offset<=(32*512)) || ( ($offset>=(105*512)) && ($offset<=(1384*512)) ) ) {
                $y="#";$n="o";
        } else {
                $y="D";$n=".";
        }
        if ($data eq $subst) {
                print "$n";
        } else {
                print "$y";
        }
        $offset+=$size; $data='initialized';
        $count=sysread IMG,$data,$size;
        if ($shown++>78) {
                printf "\n%08x ",$offset;
                $shown=0;
        }

}
print "\n\n";
close IMG;
```

Taking a look at the floppy contents

```
root@cholla# mount -t vfat -o loop,ro,noexec,noatime fl-260404-RJL1.cpy /mnt
root@cholla# catdoc /mnt/*.doc
```

Retrieving timelines

The timeline shown in the document has been created using "fls –arpl" against the image. This provides sufficient granularity for the task at hand. A full forensic timeline has also been created and is shown below, but was not used as part of the explanation.

```
daniel@cholla$ fls -f fat12 -m/ -r -zMST fl-260404-RJL1.cpy  > fls
daniel@cholla$ ils -m -f fat12 fl-260404-RJL1.cpy  > ils
daniel@cholla$ cat fls ils > mac
daniel@cholla$ mactime -b mac > timeline
daniel@cholla$ cat timeline
Sat Feb 03 2001 19:44:16    36864 m.. -rwxrwxrwx 0      0      5     <fl-260404-RJL1.cpy-_AMSHELL.DLL-dead-5>
Sun Feb 04 2001 03:44:16    36864 m.. -/-rwxrwxrwx 0     0      5     /CamShell.dll (_AMSHELL.DLL) (deleted)
Fri Apr 23 2004 01:31:06    33423 m.. -/-rwxrwxrwx 0     0      17    /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                            32256 m.. -/-rwxrwxrwx 0     0      13    /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Fri Apr 23 2004 10:53:56      727 m.. -rwxrwxrwx 0      0      28    <fl-260404-RJL1.cpy-_ndex.htm-dead-28>
Fri Apr 23 2004 19:53:56      727 m.. -/-rwxrwxrwx 0     0      28    /_ndex.htm (deleted)
Fri Apr 23 2004 20:54:32   215895 m.. -/-rwxrwxrwx 0     0      23    /Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 20:55:26   307935 m.. -/-rwxrwxrwx 0     0      20    /Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 23:10:50    22528 m.. -/-rwxrwxrwx 0     0      27    /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Fri Apr 23 2004 23:11:10    42496 m.. -/-rwxrwxrwx 0     0      9     /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Sun Apr 25 2004 09:00:00        0 .a. -/-rwxrwxrwx 0     0      3     /RJL        (Volume Label Entry)
Sun Apr 25 2004 19:53:40        0 m.c -/-rwxrwxrwx 0     0      3     /RJL        (Volume Label Entry)
Mon Apr 26 2004 00:00:00    36864 .a. -rwxrwxrwx 0      0      5     <fl-260404-RJL1.cpy-_AMSHELL.DLL-dead-5>
                              727 .a. -rwxrwxrwx 0      0      28    <fl-260404-RJL1.cpy-_ndex.htm-dead-28>
Mon Apr 26 2004 09:00:00    42496 .a. -/-rwxrwxrwx 0     0      9     /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
                            36864 .a. -/-rwxrwxrwx 0     0      5     /CamShell.dll (_AMSHELL.DLL) (deleted)
                            22528 .a. -/-rwxrwxrwx 0     0      27    /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
```

```
                          32256 .a. -/-rwxrwxrwx 0        0       13       /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
                          33423 .a. -/-rwxrwxrwx 0        0       17       /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
                         215895 .a. -/-rwxrwxrwx 0        0       23       /Remote_Access_Policy.doc (REMOTE~1.DOC)
                            727 .a. -/-rwxrwxrwx 0        0       28       /_ndex.htm (deleted)
                         307935 .a. -/-rwxrwxrwx 0        0       20       /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:18  36864 ..c -rwxrwxrwx 0          0        5       <fl-260404-RJL1.cpy-_AMSHELL.DLL-dead-5>
Mon Apr 26 2004 09:47:36    727 ..c -rwxrwxrwx 0          0       28       <fl-260404-RJL1.cpy-_ndex.htm-dead-28>
Mon Apr 26 2004 18:46:18  36864 ..c -rwxrwxrwx 0          0        5       /CamShell.dll (_AMSHELL.DLL) (deleted)
Mon Apr 26 2004 18:46:20  42496 ..c -/-rwxrwxrwx 0        0        9       /Information_Sensitivity_Policy.doc
(INFORM~1.DOC)
Mon Apr 26 2004 18:46:22  32256 ..c -/-rwxrwxrwx 0        0       13       /Internal_Lab_Security_Policy1.doc
(INTERN~1.DOC)
Mon Apr 26 2004 18:46:24  33423 ..c -/-rwxrwxrwx 0        0       17       /Internal_Lab_Security_Policy.doc
(INTERN~2.DOC)
Mon Apr 26 2004 18:46:26 307935 ..c -/-rwxrwxrwx 0        0       20       /Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 18:46:36 215895 ..c -/-rwxrwxrwx 0        0       23       /Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 18:46:44  22528 ..c -/-rwxrwxrwx 0        0       27       /Acceptable_Encryption_Policy.doc
(ACCEPT~1.DOC)
Mon Apr 26 2004 18:47:36    727 ..c -/-rwxrwxrwx 0        0       28       /_ndex.htm (deleted)
```

Extracting strings from camshell

daniel@cholla$ `strings fl_camshell.dll` to extract ASCII text strings

daniel@cholla$ `strings -el fl_camshell.dll` to extract strings using 16bit character codes

The second method is more successful, and results in the identification strings shown in the first part of the document. Especially the information about the author (Twisted Pear Productions) was instrumental in locating an original copy of the tool with Google.

Comparing evidence camshell.dll with the original

fl_camshell.dll is the file as extracted from the evidence floppy. The initial MD5 checksum shown (6462fb3acca0301e52fc4ffa4ea5eff8) allows to tie this procedure back to the original evidence at any time. camshell.dll is the file as obtained from installing Camouflage V1.2.1, the original checksum is also shown. The command "dd" was used to strip the first two blocks away from both files, given that the first two blocks of the evidence file were overwritten/corrupt. Comparing the resulting remainder of both files with MD5 results in a near certain proof that the files are indeed the same.

```
creosote:/tmp # md5 fl_camshell.dll
6462fb3acca0301e52fc4ffa4ea5eff8    fl_camshell.dll
creosote:/tmp # md5 camshell.dll
4e986ab0909d2946bed868b5f896906f    camshell.dll
creosote:/tmp # dd if=fl_camshell.dll bs=512 skip=2 of=fl_clean_camshell.dll
70+0 records in
70+0 records out
creosote:/tmp # dd if=camshell.dll bs=512 skip=2 of=fl_clean_camshell.dll
70+0 records in
70+0 records out
creosote:/tmp # md5 fl_clean_camshell.dll
aaf222265674efd802361f560f305a74    fl_clean_camshell.dll
creosote:/tmp # md5 clean_camshell.dll
aaf222265674efd802361f560f305a74    clean_camshell.dll
creosote:/tmp #
```

Reverse-engineering the password "encryption" of Camouflage

We started with camouflaging a file "a.txt" containing "00000000" within a file called "b.txt" containing "11111111", using an empty password. Taking a look at the resulting file in a Hex Editor did not reveal any tell-tale patterns. But already after hiding "a.txt" within a file "c.txt" containing "22222222", again without password, and

comparing the results did we start to suspect that the observed string near the end of the file might be a Camouflage end marker. This was quickly verified by camouflaging some bigger files and checking the results. With this finding of a common end marker of "$0x74,0xA4,0x54,0x10,0x22,0x97$", we were already able to establish that the evidence files "Internal_Lab_Security_Policy.doc", as well as "Password_Policy.doc" and "Remote_Access_Policy.doc" for sure contained camouflaged information.

As a next step, we started to try out varying passwords. First, a password of "A" was used, then "AA" and "AAA", and again the results were compared in the Hex editor. The files did not show many differences, but there was a structure near the end of the file that seemed to "grow" in sync with our longer passwords. It was quickly verified that the number of characters in that section indeed reflected the number of characters in the password. In other words, unlike the comments of the Camouflage authors in the README file accompanying the program, it seemed as if the "encrypted" files did contain a copy of the password. Encouraged by the fact that the entered password and the hidden password were of the same length and that consequently we were looking for a "substitution" type of cipher rather than something more elaborate, we started to try various passwords and to observe the effect on the password stored in the file. The two passwords that finally helped to reveal the substitution algorithm were "AAAA" and "@@@@". The character code of these two passwords only varies in one bit, and -- so did the characters of the "encrypted" passwords. We were finally able to retrieve the obfuscation values by performing an XOR calculation between the original and "encrypted" passwords
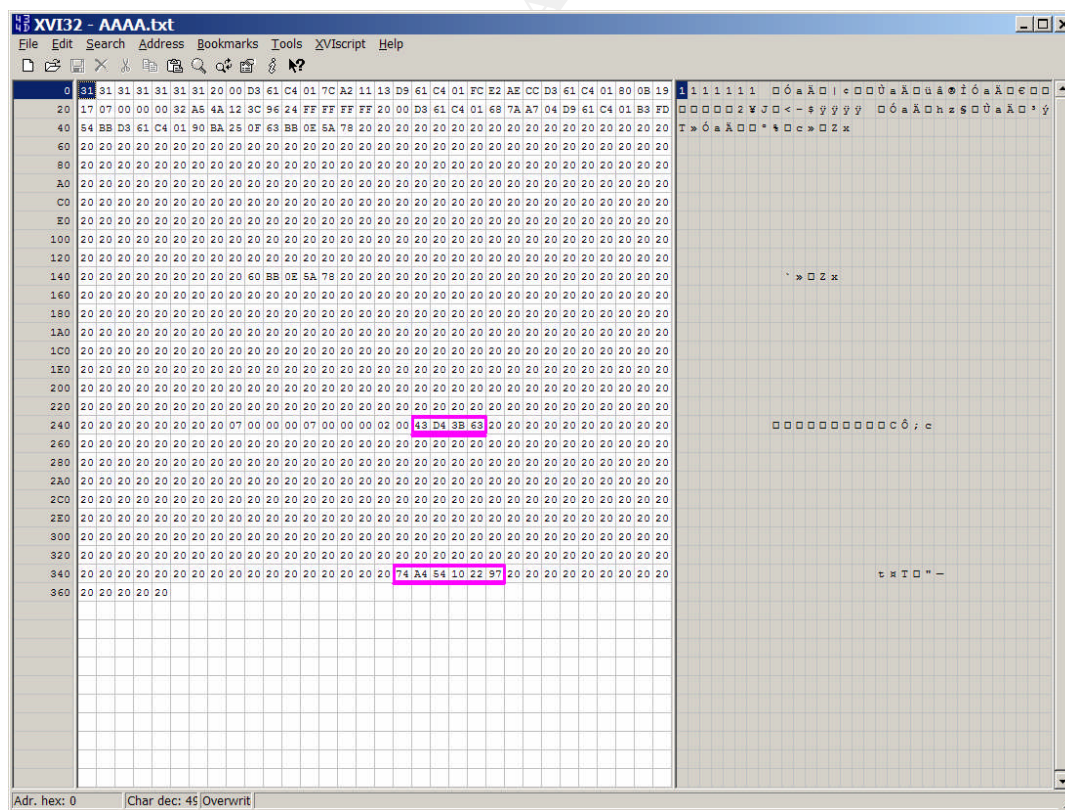
Password entered: AAAA
Character Code: 0x41 0x41 0x41 0x41
Password found in File: 0x43 0xD4 0x3B 0x63   (shown in Hexedit above)
XOR of the two: 0x02 0x95 0x7A 0x22

Following this method for a password of "AAAAAAAA", we were able to obtain the XOR values for an eight-character Password, and subsequently to decode the passwords of the files contained in the evidence. We then created the script shown below, which can be used to detect the presence of camouflaged files and to reveal the hidden passwords.

```
daniel@cholla gcfa$ cat reveal.pl
#!/usr/bin/perl
#
# Detects files containing stuff hidden with "camouflage v1.2.1"
# and also reveals the "encryption" password
#
# Reverse-Engineered and written 07Jul2004 by Daniel Wesemann

$PWDOFFSET=0xed;
$MAGICOFFSET=0x1ec;
$MAGIC=chr(0x74).chr(0xa4).chr(0x54).chr(0x10).chr(0x22).chr(0x97);
@XORVALS=(0x02,0x95,0x7A,0x22,0x0c,0xa6,0x14,0xe1);

while ($filename=shift) {
        open (CF,$filename) || die "cant find $filename";
        sysseek CF,-512,2;
        sysread CF,$buf,512;
        close CF;

        $magic=substr $buf,$MAGICOFFSET,length($MAGIC);
        if ($magic eq $MAGIC) {
                $i=0;
                $pass=substr $buf,$PWDOFFSET,scalar @XORVALS;
                $pass=~s/ .*//g;
                $pass=~s/(.)/chr(ord($1) ^ $XORVALS[$i++])/ge;
                print "[CAMO]  $filename,  password \"$pass\"\n";
        } else {
                print "[PLAIN] $filename\n";
        }
}
daniel@cholla gcfa$

daniel@cholla gcfa$ ./reveal.pl camo/*
[PLAIN] camo/Acceptable_Encryption_Policy.doc
[PLAIN] camo/Information_Sensitivity_Policy.doc
[CAMO]  camo/Internal_Lab_Security_Policy.doc,  password ""
[PLAIN] camo/Internal_Lab_Security_Policy1.doc
[CAMO]  camo/Password_Policy.doc,  password "Password"
[CAMO]  camo/Remote_Access_Policy.doc,  password "Remote"
[CAMO]  camo/testcamo.txt,  password "AAAA"
[PLAIN] camo/testplain.txt
daniel@cholla gcfa$
```

References to Tools and Documents used

XVI32   Windows Hex Editor, by Christian Maas
Helix   Bootable Forensic CD with Linux, by e-fense.com
Sleuthkit   forensic tool chest, by Brian Carrier
Perl   The Mother of all Programming Languages, by Larry Wall
Autopsy 2.0 Forensic Browser, by Brian Carrier
VMWare   my virtual lab system, by vmware.com
catdoc   a Microsoft Word "reader" for Linux, by Vitus Wagner

[URL]   Discussion thread on comp.security.pgp.discuss
             on the cryptographical strength (or lack thereof) of Camouflage
[URL]   Discussion thread on sci.crypt about ways of detecting files
             that contain camouflaged content
[URL]   How to detect XOR encoded text using the Kappa function,
             paper by Karl Frank, Salzburg (not dated)
[URL]   Swiss Federal Code of Obligations
[URL]   Swiss Federal Criminal Code

[BOOK]   The Hitchhiker's Guide to the Galaxy, by Douglas Adams
             Ballantine Books, 1980, ISBN 0-345-39180-2

Part 2 -- Forensic Analysis of a System

Synopsis of Case Facts

This is a forensic analysis of an used hard drive acquired in an online auction at eBay. The disk was deliberately acquired through a vendor who routinely seems to trade in second-hand computer hardware on eBay, to make it unlikely that the vendor was in any way related to the previous owner of the device.

The task of the engagement was threefold:

- to identify the previous owner and some of his/her habits
- to look for evidence suggesting that the system had been subverted by malware or viruses or contained a back door or sniffer programs
- to investigate whether illegal material like pirated software, sexually explicit content or music files had been used, transmitted or stored on the system under investigation, by verifying the Internet history as well as residual content on the hard drive

Of these three steps, step one was the most successful. Even though the hard disk had been reformatted before being sold on eBay, residual data still contained hundreds of recoverable Microsoft Word documents. The information encountered was exhaustive enough not only to identify the previous owners, a family living in northeastern Switzerland, but would also allow to compile a detailed list of their affiliations and habits. To protect the privacy of the three individuals, most of this information has been withheld or disguised in this paper.

An analysis of the binaries (programs) recovered from the hard drive did not reveal any evidence that the system had been subverted or infested by a virus. The operating system was identified as the German language version of Microsoft Windows 95.

While more than 1500 files containing images/pictures could be reconstructed from the hard drive, most of the files were what you would expect to find in the cache of a web browser on a family PC. None of the pictures encountered were sexually explicit. The collection of software installed on the system is extensive, including for example two different word processors, Word Perfect and Microsoft Word. Since we consider it unlikely that somebody would put up the money to *buy* two sets of word processors and end up using only one of them, we assume that most of the programs installed on the system are pirated copies. We were unable to *prove* this, though, since none of the license numbers found on the system actually showed up on a warez web site when searching with Google.

To sum up, as far as we can tell, the previous owners have not done anything wrong, except making the mistake to give away an used hard drive without properly cleaning it by using a wiping tool. Apparently, second hand hard drives are a bountiful and risk-free way to harvest personal information – a disconcerting find, given that the crime of identity theft is on the uprise world wide.

Hard disk acquisition

The hard disk was acquired for the symbolic amount of 1SFr (75cts) on April 29, 2004, through ricardo.ch, the Swiss variant of eBay. When the disk arrived by mail six days later, we tagged it and locked it away until the investigation began.

ID Tag Number	Description
TRO-20040429-01	Quantum Fireball 1280A ATA Disk drive 2/24 6072-2-009-0728 FB 1.2 GB Art.Nr. FEQU19 1280AT P/N FB12A012 Rev 01-A C/H/S 2484/16/63 Jumpers set to "slave"

While the jumpers on the disk were set to "slave", making the disk a secondary hard drive in a dual drive computer, the contents of the disk were later shown to likely be those of a primary disk drive. Our assumption is that the jumpers were set this way when the disk was briefly inserted as a secondary drive into another, unrelated system in order to re-format the drive before sale.

Image creation

The disk was connected to an old, otherwise unused PC which has been slightly customized for forensic acquisition of disk images. The PC starts from a hard disk containing a copy of SuSE Linux that has been adapted to not to touch any other hard drive during bootup. The PC connects over a direct network cable (crossover) link to the forensic analysis station which is also running SuSE Linux. The detailed log of the image acquisition process can be found in the appendix of this document. This is only a summary.

Forensic analysis is always being performed on a copy, and never on original evidence. The original hard drive needs to remain in its pristine state to ensure that no evidence is being destroyed or – even worse – introduced in the course of the analysis process. Keeping the original drive in a safe place, with an intact chain of custody log, allows to always verify the findings against the original evidence, should the need arise.

Image acquisition started by calculating the MD5 hash of the entire evidence drive. An MD5 hash is a cryptographic checksum used to verify the integrity of a file. It is generally accepted in the forensic community and by courts of law that two files or devices having the same MD5 checksum are, in fact, identical.

```
root@pc2 root]# md5sum /dev/hdd
3c744ad303c6e871f2ef4f56bb56e043  /dev/hdd
```

The entire content of the original drive was then "streamed" over the network link to the forensic analysis station, and stored into a file there. The MD5 function was used to ensure that the checksum of this file copy matched the checksum of the original device, confirming that the resulting file was indeed a true bit copy of the original evidence.

```
creosote:~/quantum # md5sum hdd.img
3c744ad303c6e871f2ef4f56bb56e043  hdd.img
```

These steps resulted in a device level copy of the original disk. While important as evidence, the entire disk is of little use for forensic analysis. Disks are usually segmented into sections by an initialization process called "partitioning". In the next step, we therefore retrieved the layout information of the original disk

```
root@pc2 root]# fdisk -l /dev/hdd

Disk /dev/hdd: 1281 MB, 1281982464 bytes
64 heads, 63 sectors/track, 621 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

   Device Boot    Start      End    Blocks   Id  System
/dev/hdd1             1      620   1249888+    7  HPFS/NTFS
```

The disk contains one partition, filling the entire disk.   Rather than to copy this partition over to the analysis station as well, we went and extracted the partition from the copy of the entire disk that had already been sent to the analysis station in the previous step. Details are shown in the appendix, what counts is the result:

```
root@pc2 root]# md5sum /dev/hdd1
6051d7407d3d5b54fe679aea68523374  /dev/hdd1

creosote:~/quantum # md5sum hdd1.split
6051d7407d3d5b54fe679aea68523374  hdd1.split
```

confirming that the original partition and the extracted copy are identical. This step concluded the evidence acquisition – the original evidence hard drive was disconnected and again locked away. All subsequent steps were performed against the two images just created.

Forensic verification

It was apparent already from the information retrieved and used during the image acquisition process that the vendor of the drive had initialized (formatted) the disk before shipping it. In fact, from the dates and timestamps retrieved, it appears as if the disk has been connected to a system running Windows XP (output of fsstat), to format the drive with a new NTFS file system structure right before shipping, on May 4, 2004, 14:03.

The screenshot on the following page, taken within the forensic tool "Autopsy", shows the only information that can be retrieved from the disk's directory. The listed system files were created when the disk was formatted on May 04, 14:03. Originally, the disk had likely contained a Windows FAT16 or FAT32 file system, but this information and all associated time lines were overwritten and destroyed.
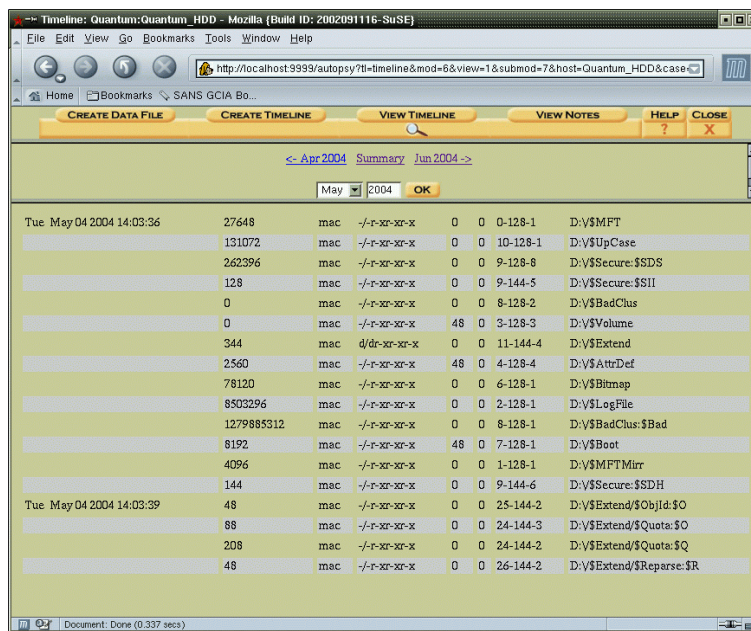
Media Analysis and Investigation

Having established that the control structures of the disk had been overwritten and the disk appeared to be empty, we started the analysis by creating an image containing only those blocks registered as unallocated. This so-called DLS extraction speeds up all subsequent analysis steps, and also helps to ensure that the original information is tainted as little as possible by the data added or changed when the disk was formatted on May 4.

```
creosote:~/quantum # dls –fntfs hdd1.split > hdd1.dls
creosote:~/quantum # md5sum hdd1.dls
f11bad7a2040cdc5283fb4f63e155fa7  hdd1.dls
```

Not knowing what to look for initially, we started with extracting text strings from the DLS image in order to verify that the unallocated sectors of the disk indeed contained residual data. The first tool used was a home-made script written to extract a fragment of everything that looked like a Microsoft Word document. From experience, we knew that hard drives from Windows PCs usually contained at least a handful of Microsoft Office documents, and that meta information contained in these documents often gave away good initial leads.

```
  68460544
R.o.o.t.
.E.n.t.r.y.........F...3...........................`.................................F.....|r.......k.............W.o.r.
d.D.o.c.u.m.e.n.t...........................o#.................................................`...........0.....
...C.o.m.p.O.b.j.........O...........F.......b..............................................................
j...u.....S.u.m.m.a.r.y.I.n.f.o.r.m.a.t.i.o.n.........................o#(.................f...........................
..............................................................................................................
..............................................................................................................
..............................................................................................................
..............................................................................................................
..............................................................................................................
.....................................................F....Microsoft.Word.Dokument.....MSWordDoc.....Word
.Document.6..9.q....................................................Oh.«..+'..0...............................
.....................................................$.......0.......X.......d.......p.......|..................
......Quartierverein.Egelshofen.................CENSORED.Georg.............................Normal.dot.........
.CENSORED.Georg..........3...........Microsoft.Word.for.Windo..D.o.c.u.m.e.n.t.S.u.m.m.a.r.y.I.n.f.o.r.m.a.t.i.o.n.......
....8.........................................................................................................
..............................................................................................................
..............................................................................................................
..............................................................................................................
..............................................................................................................
....+,.®0...............H.......P.......l.......t.......|.............................8280.Kreuzlingen.........
L......8.......................................Quartierverein.Egelshofen....................................
..............................................................................................................
```

The script output shown above contains the typical structure of the last bytes of a Microsoft Word document. Both the name of the author (last name changed to "CENSORED" for privacy reasons) and his ZIP code and city can be retrieved. We can also see the likely title of the document.

Without any substantial effort, this first finding established
- that the disk indeed contained data fragments worth investigating
- a likely candidate for the name of the system owner: Georg Censored
- two hints of where the owner might live: in 8280 Kreuzlingen, Switzerland, in a neighborhood called "Egelshofen"[2]

As a next step, we extracted all text strings of ten characters or longer.

```
creosote:~/quantum # cat hdd1.dls | strings --bytes=10 --radix=d > hdd1.strings10
```

A simple word count on the resulting collection of strings looking for the family name found above helped to confirm that "Censored" might indeed be the name of the owner of the system:

```
creosote:~/quantum # egrep -i "censored" hdd1.strings10 | wc -l
2305
```

The potential family name was found 2305 times on the disk, and the related search also uncovered plenty of email addresses containing the name.

```
creosote:~/quantum # egrep -i "censored" hdd1.strings10 | egrep "\@" | more
[…]
To: "Annemarie Censored" annemarie.censored@bluewin.ch
[…]
```

Verifying the integrity of executable files

Given that the disk had been formatted and that consequently there was no directory structure and no timeline available, this task was not easy to perform. Knowing that Microsoft executable files have a specific header structure, we wrote a small Perl script (see appendix) to hunt for fragments of executable programs in the DLS file. First attempts were successful, so we modified the script to extract all the possible .exe files from the image and to store them as individual files. This resulted in a whooping 2455 potential executables that we had to investigate further.

The National Software Reference Library (http://www.nsrl.nist.gov) contains crypto-graphic fingerprints of a lot of commercial software – but knowing from some of the un-covered text strings that the operating system on the evidence drive had likely been a German version of Windows95, we doubted that the NSRL lists would be of much help. And indeed, only nine out of our two thousand files came back with a match when compared to the NSRL Operating System and Foreign Language lists (sets A and B).

Thus, to be more certain, we obtained an original German Win95 installation CD, installed the operating system inside a VMWare machine, and then went on to compile a list of known good checksums to compare against. This approach was far

---

[2] Quartierverein Egelshofen translates into Egelshofen Neighborhood Society

more successful, a total of 85 binaries from the evidence disk were positively identified as being part of an original Windows 95 installation.

Since some of the "more important" system files like FDISK, DEBUG, EDIT, SHARE, KRNL386.EXE, IO.SYS and NBTSTAT were also part of the subset of the files that could be positively verified, we conclude that the evidence disk had very likely *not* been infected by a file-borne virus.

To be more certain about the latter, we tried to find out why the central piece of Microsoft DOS and Windows95, the command interpreter "COMMAND.COM", was not among the files that could be verified to be an original. Looking for the string "COMMAND" among the 2455 extracted files resulted in two potential candidates, the files 60729856.exe and 60852736.exe. A verification of the checksum revealed that these two files were identical, but both were with 93366 bytes length also 2016 bytes shorter than the original "COMMAND.COM" from our Windows95 test installation.

```
creosote:~/quantum/exes # ls -al command.com 60852736.exe
-rw-r--r--    1 root     root         93366 Jul 18 19:19 60852736.exe
-rwxr-xr-x    1 root     root         95382 Jul 18 18:11 command.com
creosote:~/quantum/exes # md5sum command.com 60852736.exe
8019eb5d5716b4ba52e9fa8c184db184  command.com
eced965ebeb08373c8d9bcbf5b620406  60852736.exe
```

By truncating the original Windows95 command.com to the same length as the file found on the evidence drive, we were still able to prove that the two files were closely related.

```
creosote:~/quantum/exes # dd if=command.com of=command.zap bs=1 count=93366
93366+0 records in
93366+0 records out
creosote:~/quantum/exes # ls -al command.zap 60852736.exe
-rw-r--r--    1 root     root         93366 Jul 18 19:19 60852736.exe
-rw-r--r--    1 root     root         93366 Jul 18 20:49 command.zap
creosote:~/quantum/exes # md5sum command.zap 60852736.exe
eced965ebeb08373c8d9bcbf5b620406  command.zap
eced965ebeb08373c8d9bcbf5b620406  60852736.exe
```

After truncating, the MD5 checksums of both files match.

Extracting Images

As a next step, we went to hunt for fragments of images. Since the evidence system had apparently contained many different imaging programs (CorelDraw and Photoshop, to name two), we expected to find a number of pictures of some sort on the drive.

To extract the pictures, we reverted to the forensic extraction tool "foremost". Foremost was run with the configuration file in the appendix, and ended up carving 8418 files out of the evidence disk image. Roughly 1600 of these were labeled by Foremost as being GIF or JPG images.

We then used the freeware picture viewer "JCDsee" to browse through the thumbnail versions of the retrieved images (screenshot shown below). This step did not reveal anything out of the ordinary; most of the retrieved images were apparent cached

copies of images and advertisement banners found on benign, innocuous Internet web pages. None of the pictures showed partial or full nudity or questionable content. The most frequent banners were from bluewin.ch, the biggest Internet provider in Switzerland. This finding tied in nicely with the email address of Annemarie Censored that we had uncovered earlier on, and which was also located in the "bluewin.ch" domain.
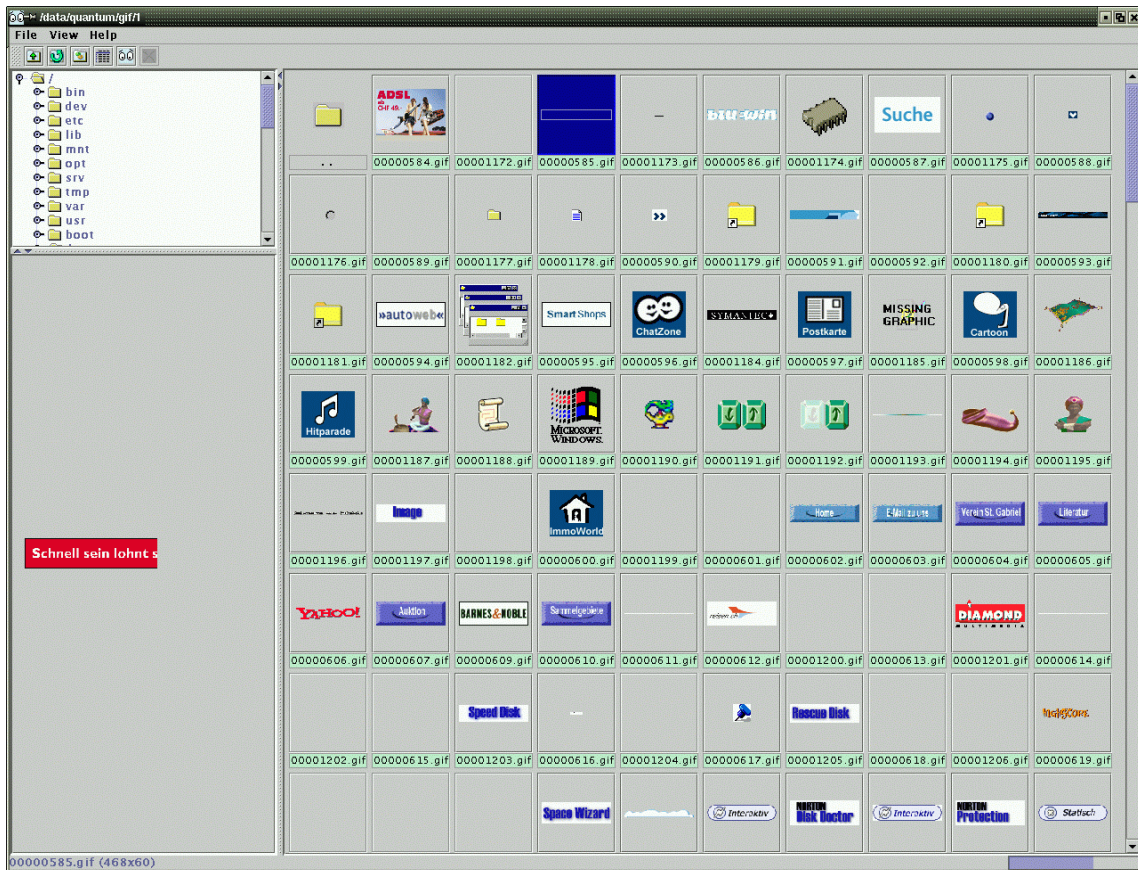
We decided to track the origin of the blue "buttons" shown on the screenshot above. Picture 00000604.gif is labeled with "Verein St. Gabriel"[3], a name conspicuous enough that a search in Google might return a good hit. And indeed, looking for "verein st.gabriel" in Google returns on third position the web page containing the blue "buttons. Pulling down the images from the web site and comparing the MD5 hash values with the evidence resulted in positive proof that the pictures indeed were the same. While this is not fully conclusive, the topic of the web page – collection of stamps with catholic religious motives – is quirky enough to assume that casual browsing would not lead anybody to this page by chance, and that somebody of the Censored family therefore has a deeper interest for stamp collection and religious topics.

---

[3] Association of Saint Gabriel

Text Documents

Foremost also extracted close to 5500 apparent Microsoft Word documents. Knowing from past experience that the Microsoft Word extraction of Foremost was nowhere near as reliable as the GIF image extraction, we went on to isolate all those files that appeared to contain a valid Microsoft Word file type marker. The one thing all true Word documents have in common[4] is the text string "Word.Document.{number}" somewhere near the end of the file. The resulting 3305 files were still too many to be true, so as a next step we retained only those files containing the family name "Censored". We had shown earlier that this name, as well as the city name Kreuzlingen, were stored in at least some of the Word documents authored by the owners of the evidence PC.

This further reduction resulted in 589 files, one of which is shown below. The text translates into an explanation of catholic church holidays for the month of February, with gaps that students have to fill in. This evidence, in addition to information uncovered earlier, strongly suggests that somebody of the Censored family is actively engaged in religious teaching. Details from some of the documents found on the evidence drive let us assume that Annemarie Censored is teaching scripture to first graders. Her husband, Georg Censored, is also a teacher, apparently instructing high school seniors in physics.

---

[4] At least the documents created with Word6, as is the case here

From looking at some of the documents with the aid of the forensic browser "Autopsy", it seemed as if what could have been the name of the original document was stored in a separate data structure within each of the files. While the data format of Word documents is too complex to allow for automatic extraction, the average document seemed to contain the interesting structure at a fixed offset from the version string "Word.Document.6" that we had mentioned earlier on. A quick Perl script was patched together to extract this information, and the outcome was moderately successful – the output at least helped to judge whether the document warranted further investigation. The script and a longer list of extracted document names can be found in the appendix.

```
creosote:/data/quantum # for i in *.doc; do ./title.pl $i; done | sed 's/name/CENSORED/i'
00000174.doc could be Abrahams - Geschichte
00000176.doc could be Georg CENSORED                    Kreuzlingen,
00000177.doc could be Abrahams – Geschichte
[Above translates to "History of Abraham"]
00000178.doc could be Georg CENSORED                    Kreuzlingen,
00000179.doc could be Georg CENSORED                    Kreuzlingen,
00000228.doc could be Die Erstkommunionfotos sind da
00000474.doc could be Die Erstkommunionfotos sind da
[Above translates to "The fotos taken at First Communion have arrived"]
00000477.doc could be In Kreuzlingen lässt sich mit wenigen Ausnahmen eigentlich nicht von
einer sehr alten Fasnachtstradition
00000479.doc could be Februar -  ein Monat mit kirchlichen Festen
00000480.doc could be Generalversammlung in Einsiedeln
00000482.doc could be Generalversammlung in Einsiedeln
00000483.doc could be Die Erstkommunionfotos sind da
00000487.doc could be Februar -  ein Monat mit kirchlichen Festen
00000488.doc could be Februar -  ein Monat mit kirchlichen Festen
00000489.doc could be Generalversammlung in Einsiedeln
00000491.doc could be Generalversammlung in Einsiedeln
00000699.doc could be Andreas CENSORED                                        10
00000700.doc could be Jahresbericht über das Geschehen im und rund um den Philatelisten-Verein
Kreuzlingen von 2001
[Above translates to "Annual Report of the Stamp Collectors Club Kreuzlingen, 2001"]
00000702.doc could be Name und Nummer
00000703.doc could be Begrüssung
00000704.doc could be Klausur  Magnetismus - Induktion – Wechselstrom
[Above translates to "Exam on magnetism, induction and alternating current"]
00000705.doc could be Protokoll der Generalversammlung in Einsiedeln
00000716.doc could be Name und Nummer
00000717.doc could be Name und Nummer
00000718.doc could be Begrüssung
00000719.doc could be Klausur  Magnetismus - Induktion - Wechselstrom
00000720.doc could be Protokoll der Generalversammlung in Einsiedeln
[Above translates to "Minutes of the general assembly in Einsiedeln"]
```

MP3 Music Files

No evidence for trading of pirated music was uncovered. Searching for MP3 files is not trivial because the files do not have a well defined start marker, but we assume that at least some audio files would contain an ID3 content description header. While a search run in comparison against a disk known to contain MP3 files turned up plenty of ID3 matches, the same search when performed against the evidence drive came up completely dry.

Email

Most of the messages were only extracted to help in establishing a time line, but we also put together a list of the "Subject:" headers of all encountered Emails. A total of

156 Email messages was found, some of the subjects are shown below. Checking the contents of some of the messages did not reveal any substantially new information – most messages had been sent or received by Annemarie Censored and dealt with either stamp collecting or religious questions. A good number of the messages were also simply personal messages between friends.

```
creosote:~/quantum # cat hdd1.strings10 | egrep '(ubject: )'
68146496 Subject: untistunden
68149148 Subject: namenspatron
132912528 Subject: Re: Ausfallzeit
139578491 Subject: =?iso-8859-1?Q?Winterst=FCrme?=
139578997 Subject: Hallo
139579988 Subject: =?iso-8859-1?Q?Valentinsgr=FCsse?=
139583186 Subject: GV
139584827 Subject: Weltenbummler
139585331 Subject: Sommerwaerme
140540774 Subject: 1.klasse CENSORED
141555013 Subject: Test
141556591 Subject: Test
141558227 Subject: Re: Mail from a mobile phone  (msg: Hallo Annemarie, ich wnsche dir
weiterhin viel Spass am Computer! Viele Grsse Patrick.)
150597207 Subject: Re: Verteilen
150598173 Subject: god jul suiza
150603977 Subject: briefmarke
154449212 Subject: Minigolf
154451307 Subject: Vorschlag.sondermarke
[…]
```

Browser History

We started this examination step with a quick check to see if the image indeed was likely to contain interesting browser history information. History files created by Microsoft Internet Explorer (IE) have a standardized start record of "Client UrlCache MMF Ver", followed by the version number. Searching for this string resulted in 30 hits, with three different version indicators, 3.2, 4.7 and 5.2, suggesting that three different versions of IE had been installed on the system.

A wide number of forensic tools (like Pasco from Foundstone) exist which allow to reconstruct the entire browsing history, but they all have in common that they need an intact browser index file ("index.dat") as input. On our evidence drive, this was not the case, as the directory structure and pointers leading to index.dat had been lost when the hard drive was reformatted. Initial attempts to piece the individual blocks together into a file that Pasco would recognize failed. We therefore reverted to writing yet another custom made Perl script (see appendix). The script, called iextractor.pl, can be run against the plain DSL image, and extracts everything that appears to be a Internet Explorer history or cache entry. An excerpt of the output is shown below. (command creosote:~/quantum # ./iextractor.pl hdd1.dls | sort)

```
01c24c42f5a9e5a0 Tue Aug  6 16:55:25 2002 Sun Aug 25 16:23:19 2002
http://adserver.bluewin.ch/gif/fleurop_234x60_summertime_d_060802.gif
01c24c42f799cec0 Sun Aug 25 00:00:07 2002 Sun Aug 25 16:23:22 2002
http://www.bluewin.ch/bw/image/0,2276,9301,00.jpg
01c24c42f94caee0 Sat Aug 24 18:50:38 2002 Sun Aug 25 16:23:25 2002
http://www.bluewin.ch/bw/image/0,2276,9497,00.jpg
01c24c42f9f487a0 Sun Aug 25 00:00:08 2002 Sun Aug 25 16:23:26 2002
http://www.bluewin.ch/bw/image/0,2276,9318,00.jpg
01c24c42fa60de00 Sun Aug 25 00:00:08 2002 Sun Aug 25 16:23:27 2002
http://www.bluewin.ch/bw/image/0,2276,9339,00.jpg
01c24c4328ef81e0 Wed Jul 31 15:41:30 2002 Sun Aug 25 16:24:45 2002
http://adserver.bluewin.ch/gif/bol_468x60_buchtipp_310702.gif
```

```
01c24c4328f72300 Sun Aug 25 16:24:45 2002 Sun Aug 25 16:24:45 2002 Visited:
annemarie@http://r.bluewin.ch/search_forwarder.fpl?fpSearchTerm=fcmuensterlingen&search_real
m=ch
01c24c4328f72300 Sun Aug 25 16:24:45 2002 Sun Aug 25 16:24:45 2002 Visited:
annemarie@http://search.bluewin.ch/bw/search/web/de/result.jsp?what=ch&qry=fcmuensterlingen
01c24c432bf21380 Sun May 26 09:45:25 2002 Sun Aug 25 16:24:50 2002
http://www.fcmuensterlingen.ch/
01c24c432cb25640 Sun May 26 09:45:11 2002 Sun Aug 25 16:24:51 2002
http://www.fcmuensterlingen.ch/logo.html
01c24c432ce4b0e0 Sun May 26 09:45:14 2002 Sun Aug 25 16:24:51 2002
http://www.fcmuensterlingen.ch/menutop.html
01c24c432cedd8a0 Sun May 26 09:45:52 2002 Sun Aug 25 16:24:52 2002
[…]
01c24cb9ef800340 Wed Jun 27 17:11:40 2001 Mon Aug 26 06:34:59 2002 http://proxy-
mss2n.bluewin.ch/bluemail.ch/suche_d.gif
01c24cb9ef800340 Wed Jun 27 17:11:42 2001 Mon Aug 26 06:34:59 2002 http://proxy-
mss2n.bluewin.ch/bluemail.ch/verschieben.gif
01c24cb9f03103c0 Mon Aug 26 06:35:00 2002 Mon Aug 26 06:35:00 2002 Visited:
annemarie@http://proxy-
mss2n.bluewin.ch/mail/Navigation?sid=32DB3366042B117B5CDA08D941214DD4EED1C892&userid=CENSORE
D@bluemail.ch&seq=+Q&auth=+A&style=de
01c24cb9f038a4e0 Mon Aug 26 06:35:00 2002 Mon Aug 26 06:35:00 2002 Visited:
annemarie@http://proxy-
mss2n.bluewin.ch/mail/MessageList?sid=32DB3366042B117B5CDA08D941214DD4EED1C892&userid=CENSOR
EDe@bluemail.ch&seq=+Q&auth=+A&srcfolder=INBOX&chk=1&style=de
01c24cb9f0496dc0 Fri Aug  9 11:17:05 2002 Mon Aug 26 06:35:00 2002
http://webmail.bluewin.ch/
01c24cb9f0496dc0 Wed Dec 12 16:10:16 2001 Mon Aug 26 06:35:00 2002 http://proxy-
mss2n.bluewin.ch/supp/bluewin.js
```

The times as reported by the "iextractor" script are first the access time in hex, to allow for easy sorting, followed by the date when the entry or the original item it refers to was last modified, followed by the time the entry was last accessed, following by the actual object or URL.

The recovered browser history starts on Friday, December 3, 1999, with an access to "ie4tour.dll/welcome.htm", the standard start page of Internet Explorer 4. Subsequent accesses are initially all against the local disk, for example to open manuals of installed software, suggesting that the Censored family maybe did not have an Internet connection before October 15, 2000, when the first "Cookie" entry was recorded by the browser. The history ends at Friday, August 30, 2002. In total, 1174 lines of browser history were recovered, and allowed to confirm earlier findings (mail.bluewin.ch as mail host, strong interest in religious contents, frequent visits to www.gabriel-gilde.ch). New findings include the obvious interest in matters related to football (soccer), as well as repeated calls to the online map service www.map24.com. For the latter, several complete URLs were found in the cache that resulted in a drawn map when called up again from within a browser. Since all these maps had apparently been used to plan trips from/to the house of the Censored family, this information is being presented here in slightly modified form for privacy reasons:

```
http://www.map24.com/map24/routing.php3?map24_sid=5c7ff41acda0f5df2535668d50323301
&newroute=1&map24_sid=5c7ff41acda0f5df2535668d50323301
&sstreet=CENSORED&szip=8280&scity=Kreuzlingen&scountry=ch&sdescription=CENSORED%0D%0ACH-
8280+Kreuzlingen
&dstreet=CENSOREDdzip=93077&dcity=Bad+Abbach&dcountry=de&ddescription=CENSORED0D%0AD-
93077+Bad+Abbach&rtype=fast&desctype=detailsmap&usecity=&maptype=JAVA&mid=LINK2&x=71&y=11
```
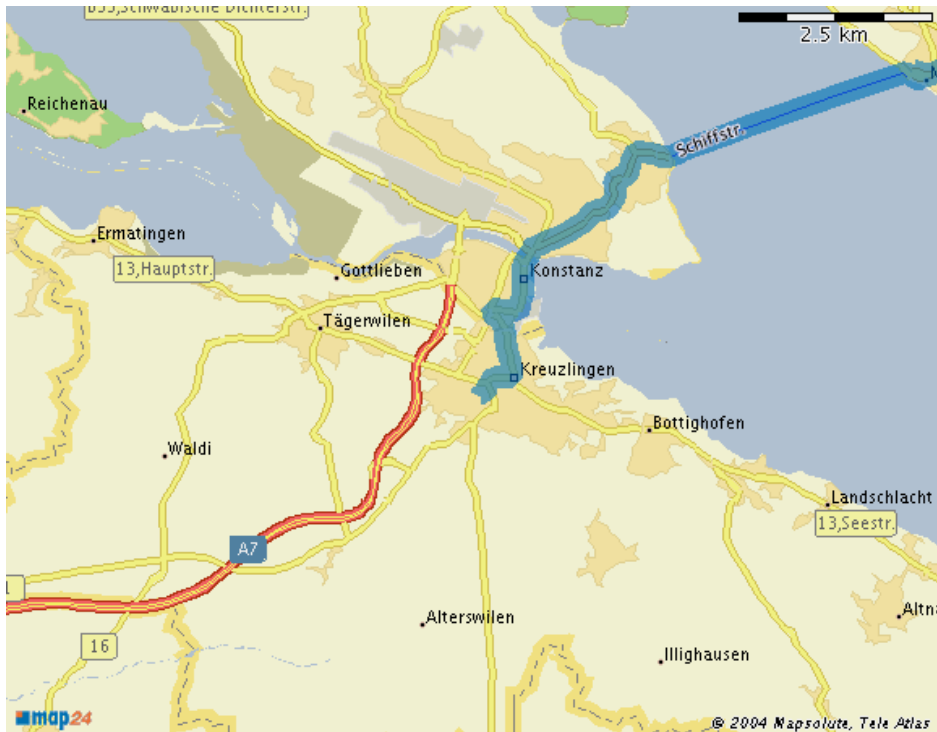
The sum of all maps (in total six) allowed us to get a very accurate insight into the relationships and holiday trips of the Censored family.

Registry Particulars

The single most interesting registry keys are those commonly abused by viruses and spyware, particularly HKLM\Software\Microsoft\Windows\CurrentVersion\Run which specifies those programs that should start automatically at system reboot. Looking for this registry key on the image resulted in quite a number of hits, but only the two matches shown below were not obvious parts of the operating system and warranted further investigation.

```
creosote:~/quantum # egrep –C2 -i "currentversion\\\run" hdd1.strings10

246985749 [InControl.Install.Reg]
246985774
;HKLM,\SOFTWARE\Microsoft\Windows\CurrentVersion\Run,DiamondRun,4,%InControl.dir%\dmhkey.exe;
246985869 HKLM,\SOFTWARE\Diamond\Installed\%InControl.sec%,Installed,,1;
246985933 HKLM,\SOFTWARE\Diamond\Installed\%InControl.sec%,Version,,4.0;
238972657 HKCU,"keyboard layout"
238972681 HKCU,"keyboard layouts"
238972706 HKLM,"SOFTWARE\Microsoft\Windows\CurrentVersion\run","internat.exe"
238972777 [EBD_Keyboard2]
238972794 CopyFiles = EBD_CopyThem2
```

Using the forensic browser "autopsy" to take a look at the corresponding sectors of the raw evidence image allowed us to tentatively identify the first match as being part of "Diamond Multimedia Hotkey Handler", and as such most likely part of the display driver. The second match was pointing towards "internat.exe", which can be both, a benign part of the operating system that allows to switch between keyboard mappings – but also a virus like VBS/Wisis-A. Again, a check with autopsy helped to

verify that the contents of the corresponding disk block very much looked like a keyboard driver and not at all like a VBS virus.

Other interesting registry keys include those that specify the start and search page of the browser (HKLM\Software\Microsoft\Internet Explorer\Main). No evidence of any spyware or malicious browser "helper" objects could be uncovered – the only registry keys that were uncovered contained the default settings of the German version of Internet Explorer 4, with www.msn.de as default and search page.

It is worth mentioning, though, that due to the complex structure of registry files, mere string searches for registry contents are rarely conclusive. In order to use more advanced techniques like importing evidence files into a Microsoft registry editor, intact registry files would be necessary. An attempt to merge several disk blocks into a file that a registry editor would recognize failed.

As a last registry related check, we retrieved all the class identifiers that were present on the image. Class identifiers are used by Microsoft ActiveX to uniquely identify a piece of software or a module thereof.

```
creosote:~/quantum # cat hdd1.strings10 | perl -ne 's/clsid[:\\\{]{1,2}(.{8}-.{4}-.{4}-.{4}-
.{12})/print "CLSID:".uc($1)."\n" unless $seen{uc($1)}++/ie' | sort

CLSID:00000010-0000-0010-8000-00AA006D2EA4
CLSID:00000011-0000-0010-8000-00AA006D2EA4
[…]
CLSID:FF393560-C2A7-11CF-BFF4-444553540000
CLSID:FF76CB60-2EC8-101B-B02E-04021C009402
```

While a number of component identifiers were retrieved, it seems as if no corresponding dictionary is available on the Internet. Translating these IDs to an associated component name would require a large number of manual searches through Google or similar, and was therefore not performed. Clearly, class identifiers would make an useful addition to the National Software Reference Library (NSRL), since they could help to identify malicious code and installed software even on an almost completely destroyed system.

Virus Check

Since so far no evidence of malicious code or wrongdoing had been uncovered, we used the trial version of Kaspersky Antivirus Scanner for Linux to double-check on all the executable files and documents that had been extracted in the previous steps. No viruses were found.

```
creosote:/opt/kav/bin # ./kavscanner ~/quantum/exes/
```

Timeline Analysis

Since the date and time information of the individual files had been destroyed when the disk was reformatted on May 4, 2004, 14:03, reconstructing a detailed timeline of earlier events was difficult. In addition to the browser history (see above), what helped most were the various log files containing time stamps. One of them, a modem dialup log, is shown below.

```
creosote:~/quantum # cat hdd1.strings10 | perl -ne 'if (/\d{1,2}(\-
|\/|\.)\d{1,2}\1\d{4}\s\s?\d{1,2}:\d{2}:\d{2}/) {s/^\d*\s//; print $_}' >/tmp/datetime.txt

creosote:~/quantum # cat /tmp/datetime.txt
11-24-2000 18:34:24.86 - Modem type: ZyXEL Omni56K
11-24-2000 18:34:24.86 - Modem inf path: OEM2.INF
11-24-2000 18:34:24.86 - Modem inf section: OM90
11-24-2000 18:34:25.41 - 115200,N,8,1
11-24-2000 18:34:26.08 - 115200,N,8,1
11-24-2000 18:34:26.08 - 115200,N,8,1
11-24-2000 18:34:26.33 - Initializing modem.
11-24-2000 18:34:26.33 - Send: AT&FE0X4<cr>
11-24-2000 18:34:26.35 - Recv: AT&FE0X4<cr>
11-24-2000 18:34:26.35 - Recv: <cr><lf>OK<cr><lf>
11-24-2000 18:34:26.35 - Interpreted response: Ok
11-24-2000 18:34:26.35 - Send: ATV1&D2&C1S0=0<cr>
11-24-2000 18:34:26.36 - Recv: <cr><lf>OK<cr><lf>
11-24-2000 18:34:26.36 - Interpreted response: Ok
11-24-2000 18:34:26.36 - Send: ATS7=60S50=0L4M1&K3*E0&K4&H3B0S41.4=0<cr>
11-24-2000 18:34:26.39 - Recv: <cr><lf>OK<cr><lf>
11-24-2000 18:34:26.39 - Interpreted response: Ok
11-24-2000 18:34:26.43 - Dialing.
11-24-2000 18:34:26.43 - Send: ATDP##########<cr>
```
[…]

All in all, the date entries of the dialup log were spanning a time between October 12, 2000, and January 13, 2001. This ties in nicely with our earlier finding of the first browser cache entry dating from October 15, 2000, suggesting that the Censored family indeed did not have Internet connectivity before that date. What happened after January 13, 2001 is unclear, since the browser history items clearly continue beyond this date – most likely, the dialer or browser was upgraded around that time and the connection log was turned off in the process. During the time where the log was still active, main Internet usage time of the Censored family was around noon, during the evening, and frequently also until well after midnight.

When suppressing all the dialup log entries from the datetime.txt file, some more interesting entries showed up:

```
creosote:~/quantum # grep CLEANUP /tmp/datetime.txt
CLEANUP Log started at 07/11/2002 23:05:10
CLEANUP Log started at 07/11/2002 23:28:53
[…]
CLEANUP Log started at 08/29/2002 17:33:45
CLEANUP Log started at 08/29/2002 17:43:27
```

The "Cleanup Log" entries are produced by the Email program "Outlook Express" when compressing its database, an indication that we should also look for an Outlook Mailbox. Again, as before with the browser cache, the last entry is from the last days of August 2002, confirming our earlier theory that the system has not been used since.

While six potential Outlook Express mail files were recovered with the aid of the forensic extraction tool "Foremost", the resulting DBX files were too damaged to open them with a mail program. We therefore reverted to manual extraction of certain typical strings that can be found in the header of Email messages.

```
creosote:~/quantum # cat hdd1.strings10 | egrep '(eceived: |    id )'
[…]
222396935 Received: from mta6n.bluewin.ch (172.21.1.235) by mss2n.bluewin.ch (Bluewin AG 6.0.053)
222397024       id 3D0EFC7E002678FB for annemarie.CENSORED@bluewin.ch; Mon, 24 Jun 2002 01:56:17 +0200
222397120 Received: from mx0.gmx.net (213.165.64.100) by mta6n.bluewin.ch (Bluewin AG 6.0.053)
222397206       id 3D140AF90006CB6E for annemarie.CENSORED@bluewin.ch; Mon, 24 Jun 2002 01:56:16 +0200
222397302 Received: (qmail 14310 invoked by uid 0); 23 Jun 2002 23:56:17 -0000
```

```
222403799 Received: from mta16n.bluewin.ch (172.21.1.225) by mss2n.bluewin.ch (Bluewin AG 6.0.053)
222403889        id 3D3F927A0018BBE6 for annemarie.CENSORED@bluewin.ch; Mon, 29 Jul 2002 12:30:03 +0200
222403985 Received: from mx0.gmx.net (213.165.64.100) by mta16n.bluewin.ch (Bluewin AG 6.5.026)
222404072        id 3D34382A003265FA for annemarie.CENSORED@bluewin.ch; Mon, 29 Jul 2002 12:30:03 +0200
[…]
```

The time stamps from the mail headers were again extracted, and ranged from November 2001 up to August 26, 2002.

All in all, the uncovered time stamps would allow to compile an extensive list of when a member of the Censored family had been using the computer, and also whether they had been connected to the Internet during the time. While this evidence is nowhere near as precise as a file system timeline would have been, we were nevertheless surprised by the extent of the timing information that can be retrieved from a reformatted hard drive.

Conclusion

A wealth of information on the Censored family was uncovered in the course of this investigation, including facts on habits, jobs, hobbies and personal data. The analysis also uncovered the access credentials of the email account of Annemarie Censored[5]. If she hasn't changed her password since (and few users do), this would allow us to read all her email from this day onward, and to even send messages in her name. Together with the personal information like birthday date and maiden name that we were able to retrieve, the sum of our findings would allow devastating identity theft attacks against the Censored family. Luckily, identity thieves so far seem to lack either the skill or patience to acquire used hard drives through eBay and to use them as source of information. Nevertheless, we consider it a folly to sell an used hard drive in an online auction for as little as one Swiss Franc as was the case here. The potential damage by far outweighs the puny short term financial gain. Used hard drives should be physically destroyed and recycled (if they are too small to be of any use nowadays) or should be properly wiped (overwritten) before being sold.

During the investigation, we did not find any evidence of wrongdoing by the Censored family. Due to the quantity of high-priced software installed on the system, we assume that the family may be using pirated copies of some of the programs, but were unable to prove this for certain. We also did not encounter any evidence that would suggest that the system had been subverted by malware, viruses or backdoors.

As part of the analysis process, we were able to write two small tools, "exetractor.pl" and "iextractor.pl", which turned out to be very useful and which will be re-used and developed further in the course of future investigations.

The evidence drive and associated image copies, as well as any extracted documents and programs, will be retained only until this paper has been accepted and graded, and will then be professionally and irrevocably destroyed.

Zurich, July 30, 2004
Daniel Wesemann

---

[5] This information – including the search terms used - is deliberately not shown in the narrative

Appendix to Part #2

Image acquisition

Forensic analysis system	Image acquisition station
Script started on Wed Jul 14 10:16:22 2004	Script started on Wed 14 Jul 2004 10:01:18 AM CEST
	root@pc2 root]# md5sum /dev/hdd
	3c744ad303c6e871f2ef4f56bb56e043 /dev/hdd
creosote:~/quantum # netcat -l -p 12345 > hdd.img	
	root@pc2 root]# dd if=/dev/hdd | netcat -w1 192.168.16.3 12345
	2503872+0 records in
creosote:~/quantum # md5sum hdd.img	2503872+0 records out
3c744ad303c6e871f2ef4f56bb56e043 hdd.img	
creosote:~/quantum # mmls -t dos hdd.img	
DOS Partition Table	root@pc2 root]# fdisk -l /dev/hdd
Units are in 512-byte sectors	
	Disk /dev/hdd: 1281 MB, 1281982464 bytes
Slot Start End	64 heads, 63 sectors/track, 621 cylinders
Length Description	Units = cylinders of 4032 * 512 = 2064384 bytes
00: ----- 0000000000 0000000000	
0000000001 Primary Table (#0)	Device Boot Start End Blocks
01: ----- 0000000001 0000000062	Id System
0000000062 Unallocated	/dev/hdd1 1 620 1249888+
02: 00:00 0000000063 0002499839	7 HPFS/NTFS
0002499777 NTFS (0x07)	
creosote:~/quantum # dd if=hdd.img bs=512 skip=63 of=hdd1.split count=2499776	
2499776+0 records in	
2499776+0 records out	
	root@pc2 root]# md5sum /dev/hdd1
	6051d7407d3d5b54fe679aea68523374 /dev/hdd1
creosote:~/quantum # md5sum hdd1.split	
6051d7407d3d5b54fe679aea68523374 hdd1.split	
	root@pc2 root]# exit
	Script done on Wed 14 Jul 2004 10:47:25 AM CEST
creosote:~/quantum # fsstat -f ntfs hdd1.split	
FILE SYSTEM INFORMATION	
--	
File System Type: NTFS	
Volume Serial Number: C0D5DCBBC0D5DCBB	
Volume Name: New Volume	
Version: **Windows XP**	
META-DATA INFORMATION	
--	
Range: 0 - 26	
Root Directory: 5	
CONTENT-DATA INFORMATION	
--	
Sector Size: 512	
Cluster Size: 2048	
Total Cluster Range: 0 - 624943	
Attribute Defs (MFT Entry: 4)	
$STANDARD_INFORMATION: 16	
$ATTRIBUTE_LIST: 32	
$FILE_NAME: 48	

```
$OBJECT_ID: 64
$SECURITY_DESCRIPTOR: 80
$VOLUME_NAME: 96
$VOLUME_INFORMATION: 112
$DATA: 128
$INDEX_ROOT: 144
$INDEX_ALLOCATION: 160
$BITMAP: 176
$REPARSE_POINT: 192
$EA_INFORMATION: 208
$EA: 224
$LOGGED_UTILITY_STREAM: 256
creosote:~/quantum # mkdir mnt
creosote:~/quantum # mount -t ntfs -o
loop,ro,noexec,noatime hdd1.split mnt
creosote:~/quantum # ls -al mnt
total 4
dr-x------    1 root     root         4096
May  4 14:03 .
drwxr-xr-x   3 root     root          224
Jul 14 10:52 ..
creosote:~/quantum # umount mnt
creosote:~/quantum # exit
Script done on Wed Jul 14 10:54:13 2004
```

Extracted timeline

This time line is of little consequence, as it only shows the information written to the disk when the hard drive was reformatted with NTFS prior to shipping.

```
Tue May 04 2004 14:03:36       144 mac -/-r-xr-xr-x 0         0       9-144-6  /$Secure:$SDH
                                 0 mac -/-r-xr-xr-x 0         0       8-128-2  /$BadClus
                         1279885312 mac -/-r-xr-xr-x 0         0       8-128-1  /$BadClus:$Bad
                             27648 mac -/-r-xr-xr-x 0         0       0-128-1  /$MFT
                           8503296 mac -/-r-xr-xr-x 0         0       2-128-1  /$LogFile
                            131072 mac -/-r-xr-xr-x 0         0      10-128-1  /$UpCase
                               344 mac d/dr-xr-xr-x 0         0      11-144-4  /$Extend
                            262396 mac -/-r-xr-xr-x 0         0       9-128-8  /$Secure:$SDS
                              2560 mac -/-r-xr-xr-x 48        0       4-128-4  /$AttrDef
                               128 mac -/-r-xr-xr-x 0         0       9-144-5  /$Secure:$SII
                              8192 mac -/-r-xr-xr-x 48        0       7-128-1  /$Boot
                              4096 mac -/-r-xr-xr-x 0         0       1-128-1  /$MFTMirr
                                 0 mac -/-r-xr-xr-x 48        0       3-128-3  /$Volume
                             78120 mac -/-r-xr-xr-x 0         0       6-128-1  /$Bitmap
Tue May 04 2004 14:03:39       208 mac -/-r-xr-xr-x 0         0      24-144-2
/$Extend/$Quota:$Q
                                88 mac -/-r-xr-xr-x 0         0      24-144-3
/$Extend/$Quota:$O
                                48 mac -/-r-xr-xr-x 0         0      26-144-2
/$Extend/$Reparse:$R
                                48 mac -/-r-xr-xr-x 0         0      25-144-2
/$Extend/$ObjId:$O
```

Extracting files with Foremost

The following config file was used (comment lines supressed to save space)

```
        gif     y       155000000 \x47\x49\x46\x38\x37\x61     \x00\x3b
        gif     y       155000000 \x47\x49\x46\x38\x39\x61     \x00\x00\x3b
        jpg     y       200000000 \xff\xd8\xff\xe0\x00\x10     \xff\xd9
        doc     y       12500000  \xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00
                                  \xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00 NEXT
        doc     y       12500000  \xd0\xcf\x11\xe0\xa1\xb1
        pst     y       400000000 \x21\x42\x4e\xa5\x6f\xb5\xa6
        ost     y       400000000 \x21\x42\x44\x4e
        dbx     y       4000000   \xcf\xad\x12\xfe\xc5\xfd\x74\x6f
        idx     y       4000000   \x4a\x4d\x46\x39
        mbx     y       4000000   \x4a\x4d\x46\x36
```

```
wpc        y       100000    ?WPC
htm        n       50000     <html                                    </html>
pdf        y       5000000   %PDF    %EOF\x0d      REVERSE
dat        y       4000000   regf
dat        y       4000000   CREG
```

Extracting names of text documents

```perl
#!/usr/bin/perl
# Assumes that document name is at a fixed offset from the version string
# Doesn't work on newer documents, custom made for Word 6

open (IMG,"$ARGV[0]") || die "oops";
while (sysread IMG,$data,512)  {
        if ($data=~/Word\.Document\.\d.{247}([^\x00]*)/) {
                $match=$1;
                next unless ($match=~/[0-9A-Za-z]{4}/);
                printf "$ARGV[0] could be $match\n";
                close IMG; exit;
        }
}
close IMG;


creosote:/data/quantum # for i in *.doc; do ./title.pl $i; done | sed 's/name/CENSORED/i'
00000036.doc could be Vorsommerfest der Steighexen
00000060.doc could be Vorsommerfest der Steighexen
00000061.doc could be Liebe Erstkommunionkinder
00000062.doc could be Quartierverein Egelshofen
00000063.doc could be Vorsommerfest der Steighexen
00000064.doc could be Liebe Erstkommunionkinder
00000065.doc could be Quartierverein Egelshofen
00000086.doc could be Werk
00000088.doc could be Annemarie CENSORED Kreuzlingen, im Mai 2002
00000090.doc could be Werk
00000092.doc could be Annemarie CENSORED Kreuzlingen, im Mai 2002
00000093.doc could be 1952   war   ein   gutes   jahr
00000147.doc could be 1952   war   ein   gutes   jahr
00000148.doc could be Versteht der Marcus nichts
00000149.doc could be 1952   war   ein   gutes   jahr
00000150.doc could be Versteht der Marcus nichts
00000174.doc could be Abrahams - Geschichte
00000176.doc could be Georg CENSORED                        Kreuzlingen,
00000177.doc could be Abrahams - Geschichte
00000178.doc could be Georg CENSORED                        Kreuzlingen,
00000179.doc could be Georg CENSORED                        Kreuzlingen,
00000228.doc could be Die Erstkommunionfotos sind da
00000474.doc could be Die Erstkommunionfotos sind da
00000477.doc could be In Kreuzlingen lässt sich mit wenigen Ausnahmen eigentlich nicht von einer sehr alten
Fasnachtstradition
00000479.doc could be Februar -  ein Monat mit kirchlichen Festen
00000480.doc could be Generalversammlung in Einsiedeln
00000482.doc could be Generalversammlung in Einsiedeln
00000483.doc could be Die Erstkommunionfotos sind da
00000484.doc could be In Kreuzlingen lässt sich mit wenigen Ausnahmen eigentlich nicht von einer sehr alten
Fasnachtstradition
00000486.doc could be In Kreuzlingen lässt sich mit wenigen Ausnahmen eigentlich nicht von einer sehr alten
Fasnachtstradition
00000487.doc could be Februar -  ein Monat mit kirchlichen Festen
00000488.doc could be Februar -  ein Monat mit kirchlichen Festen
00000489.doc could be Generalversammlung in Einsiedeln
00000490.doc could be Generalversammlung in Einsiedeln
00000491.doc could be Generalversammlung in Einsiedeln
00000699.doc could be Andreas CENSORED                                              10
00000700.doc could be Jahresbericht über das Geschehen im und rund um den Philatelisten-Verein Kreuzlingen von 2001
00000702.doc could be Name und Nummer
00000703.doc could be Begrüssung
00000704.doc could be Klausur  Magnetismus - Induktion - Wechselstrom
00000705.doc could be Protokoll der Generalversammlung in Einsiedeln
00000715.doc could be Jahresbericht über das Geschehen im und rund um den Philatelisten-Verein Kreuzlingen von 2001
00000716.doc could be Name und Nummer
00000717.doc could be Name und Nummer
00000718.doc could be Begrüssung
00000719.doc could be Klausur  Magnetismus - Induktion - Wechselstrom
00000720.doc could be Protokoll der Generalversammlung in Einsiedeln
00000721.doc could be Andreas CENSORED
00000897.doc could be Annemarie CENSORED
00000898.doc could be In Italien liegt eine kleine Stadt, die Assisi heisst
00000899.doc could be PHILATELISTEN - VEREIN  KREUZLINGEN
00000900.doc could be Gilde St
00000901.doc could be Gilde St
00000902.doc could be Annemarie CENSORED
00000903.doc could be In Italien liegt eine kleine Stadt, die Assisi heisst
00000904.doc could be PHILATELISTEN - VEREIN  KREUZLINGEN
00000905.doc could be Gilde St
00000906.doc could be Gilde St
00001099.doc could be Philatelisten Verein
          Kreuzlingen, 4
00001100.doc could be Jesus lebt
00001102.doc could be Klausur Magnetismus - Induktion - Wechselstrom
00001103.doc could be Klausur Bremsen
00001106.doc could be Jesus lebt
```

```
00001107.doc could be Klausur Bremsen
00001108.doc could be Klausur Magnetismus - Induktion - Wechselstrom
00001109.doc could be Klausur Bremsen
00001216.doc could be Gilde St
00001228.doc could be Annemarie CENSORED                                    Kreuzlingen, 26
00001229.doc could be Gilde St
00001527.doc could be  NAME
00001538.doc could be  NAME
00001609.doc could be Spielplan 2002/ 2003
00001610.doc could be Spielplan 2002/ 2003
00001658.doc could be Magnet - Hochspannungs - Kondensatorzündung MHKZ
00001665.doc could be Protokoll der Vorstandsitzung vom 13
00001666.doc could be Klausur Kraftübertragung
00001668.doc could be Klausur Kraftübertragung
00001669.doc could be Gott, unser Vater,
00001670.doc could be Klausur lösbare Verbindungen
00001673.doc could be Die Feier der Versöhnung
00001676.doc could be Werbung aus dem
00001677.doc could be Magnet - Hochspannungs - Kondensatorzündung MHKZ
00001678.doc could be Magnet - Hochspannungs - Kondensatorzündung MHKZ
00001679.doc could be Gilde St
00001680.doc could be Gilde St
00001681.doc could be Protokoll der Vorstandsitzung vom 1
00001685.doc could be Protokoll der Vorstandsitzung vom 13
00001686.doc could be Klausur Kraftübertragung
00001688.doc could be Klausur Kraftübertragung
00001689.doc could be Gott, unser Vater,
00001690.doc could be Klausur lösbare Verbindungen
00001691.doc could be Klausur Magnetzündung
00001692.doc could be Klausur Magnetzündung
00001693.doc could be Die Feier der Versöhnung
00001694.doc could be Klausur Magnetzündung
00001695.doc could be Werbung aus dem
00001696.doc could be Werbung aus dem
00001861.doc could be Damals brachten viele Völker Gott Menschenopfer dar
00001862.doc could be Die Botschaft zieht Kreise
00001865.doc could be Vorstandsitzung vom 29
00001866.doc could be Vorstandsitzung vom 11
00001867.doc could be Gott ist mit uns im Kirchenjahr
00001868.doc could be Damals brachten viele Völker Gott Menschenopfer dar
00001869.doc could be Die Botschaft zieht Kreise
00001872.doc could be Vorstandsitzung vom 29
00001873.doc could be Vorstandsitzung vom 11
00001874.doc could be Gott ist mit uns im Kirchenjahr
00001917.doc could be Jahresbericht über das Geschehen im und rund um den Philatelisten - Verein Kreuzlingen
00001918.doc could be  Bussfeier für 3
00001919.doc could be Zum Jahresbeginn      365 Schlüssel im Jahr meinen Händen anvertraut
00001921.doc could be  Bussfeier für 3
00001922.doc could be Zum Jahresbeginn      365 Schlüssel im Jahr meinen Händen anvertraut
00001967.doc could be Firma E
00002029.doc could be Haus "zum Gut"
00002031.doc could be Haus "zum Gut"
00002032.doc could be Klausur Wärmebehandlungen von Stahl
00002093.doc could be Klausur Wärmebehandlungen von Stahl
00002282.doc could be                          EXKURSION
00002283.doc could be DAHEIM
00002285.doc could be Der Heilige Ulrich und der Fisch
00002286.doc could be Der Heilige Ulrich und der Fisch
00002307.doc could be ERFOLGSRECHNUNG für den RUNDSENDEVERKEHR des
00002309.doc could be Kreuzlingen, Briefmarkenbörse
00002310.doc could be Rundsendedienst Gilde St
00002311.doc could be Rundsendereglement
00002312.doc could be HEILIGE UND NAMENSPATRONE
00002313.doc could be Hallo zusammen
00002314.doc could be Protokoll der Vorstandsitzung vom 17
00002316.doc could be Exkursion
00002319.doc could be KIRCHEN UND KLÖSTER
00002320.doc could be                                 Anmeldung
00002321.doc could be Fanclubturniere 1998
00002322.doc could be FCK FANCLUB  "BODENSEETEUFEL"
00002324.doc could be Gilde St
00002325.doc could be Rundsendeliste  Juli 2000
00002326.doc could be Rundsendeliste  Juli 2000
00002328.doc could be ERFOLGSRECHNUNG für den RUNDSENDEVERKEHR des
00002330.doc could be Kreuzlingen, Briefmarkenbörse
00002331.doc could be Rundsendedienst Gilde St
00002332.doc could be Rundsendereglement
00002333.doc could be HEILIGE UND NAMENSPATRONE
00002334.doc could be Hallo zusammen
00002335.doc could be Protokoll der Vorstandsitzung vom 17
00002336.doc could be Exkursion
00002340.doc could be KIRCHEN UND KLÖSTER
00002342.doc could be Fanclubturniere 1998
00002343.doc could be FCK FANCLUB  "BODENSEETEUFEL"
00002344.doc could be Gilde St
00002345.doc could be Gilde St
00002346.doc could be Rundsendeliste  Juli 2000
00002347.doc could be Rundsendeliste  Juli 2000
00004147.doc could be les Memo
00004620.doc could be Kreuzlingen Briefmarkenbörse
```

Extracting Executables

Extracting EXEs and comparing the results with reference MD5 sums obtained from a clean installation (the reference MD5 sums are available through my web page)

```
Script started on Sun Jul 18 19:18:47 2004

creosote:~/quantum/exes # ../exetractor.pl ../hdd1.dls
Found possible EXE at 0002517504 Rem: 0090  Pages: 0002  Size: 00001168
Found possible EXE at 0003287552 Rem: 0090  Pages: 0002  Size: 00001168
[…]
Found possible EXE at 1268078080 Rem: 00a2  Pages: 0000  Size: 00000162
Found possible EXE at 1270175232 Rem: 0090  Pages: 0002  Size: 00001168

creosote:~/quantum/exes # md5sum *.exe | /data/rds/win95/suck.pl
Adding 100010496.exe with 3defa7163c303ade453893b5679d3939
Adding 100248064.exe with faf5303606e674dd2ea152a4391b3b90
[…]

Collision: 115821056.exe and 110889472.exe
Adding 115976704.exe with 038b0c60f34712f7970bec26284e2883
[…]
Adding 99969536.exe with 967c641ba71c85d3a97c25d912d89939
Adding 99977728.exe with dbdaad04a8912dc7f10f2f7ba0279116

Now starting comparision with reference…


11995648.exe    could be *./DOS/SHARE.EXE
853693952.exe   could be *./DOS/SMARTDRV.EXE
247646720.exe   could be *./DOS/SCANDISK.EX~
247556608.exe   could be *./WINDOWS/COMMAND/DRVSPACE.BIN
247646720.exe   could be *./WINDOWS/COMMAND/SCANDISK.EXE (secondary match)
29567488.exe    could be *./WINDOWS/COMMAND/FDISK.EXE
60951040.exe    could be *./WINDOWS/COMMAND/EDIT.COM
29542912.exe    could be *./WINDOWS/COMMAND/DEBUG.EXE
18745856.exe    could be *./WINDOWS/COMMAND/FC.EXE
3787264.exe     could be *./WINDOWS/COMMAND/FIND.EXE
18565632.exe    could be *./WINDOWS/COMMAND/MEM.EXE
21932544.exe    could be *./WINDOWS/COMMAND/SHARE.EXE
20015616.exe    could be *./WINDOWS/COMMAND/SUBST.EXE
3426816.exe     could be *./WINDOWS/COMMAND/XCOPY.EXE
251914752.exe   could be *./WINDOWS/SYSTEM/DSKMAINT.DLL
101337600.exe   could be *./WINDOWS/SYSTEM/COMMCTRL.DLL
[…]

248515072.exe   could be *./WINDOWS/SYSTEM/SYSDM.CPL
35834368.exe    could be *./WINDOWS/SYSTEM/GDI.EXE
33450496.exe    could be *./WINDOWS/SYSTEM/KRNL386.EXE
248351232.exe   could be *./WINDOWS/SYSTEM/REDIRECT.MOD
249031168.exe   could be *./WINDOWS/SYSTEM/KEYBOARD.DRV
[…]
116763136.exe   could be *./WINDOWS/SYSTEM/COMMDLG.DLL
116853248.exe   could be *./WINDOWS/SYSTEM/SHELL.DLL
250448384.exe   could be *./WINDOWS/SYSTEM/NETCPL.CPL
211945984.exe   could be *./WINDOWS/SYSTEM/MCIPIONR.DRV
[…]
853693952.exe   could be *./WINDOWS/SMARTDRV.EXE (secondary match)
84593152.exe    could be *./WINDOWS/NDISLOG.TXT (secondary match)
3320320.exe     could be *./WINDOWS/WINHELP.EXE
65145344.exe    could be *./WINDOWS/DBLBUFF.SYS
3418624.exe     could be *./WINDOWS/WINVER.EXE
[…]
84593152.exe    could be *./WINDOWS/WIN386.SWP (secondary match)
130288128.exe   could be *./WINDOWS/WINPOPUP.EXE
48105984.exe    could be *./WINDOWS/NBTSTAT.EXE
64637440.exe    could be *./IO.SYS
creosote:~/quantum/exes # exit

Script done on Sun Jul 18 19:28:54 2004
```

The scripts used

```perl
#!/usr/bin/perl
#
# exetractor.pl - Extracts Microsoft Executables from a Forensic DLS image
# Written 18 July 2004 by Daniel Wesemann
# You can freely use and adapt this script, but please retain the
# above credits for my clever original work :-)
#
# ----------------------------------------------------------------------------

&usage if (@ARGV < 1);
open (IMG,"$ARGV[0]") || die "oops, cant read $ARGV[0]";

$offs=0;
while (sysread IMG,$data,512)  {
        if ($data=~/^MZ(.{4})/) {
                    printf "Found possible EXE at %010i ",$offs;
                    my $rem=unpack ("S2",$1);
                    my $pag=unpack ("x2S2",$1);
                    if ($pag > 0) {$pag--};
                    my $siz=$pag*512+$rem;
                    printf "Rem: %04x  Pages: %04x  Size: %08i\n",$rem,$pag,$siz;
                    if ($rem<512) {
                                if ($storing) {
                                            # we are already writing, close former first
                                            close STORE;
                                            rename $fname, "$fname.truncated";
                                }
                                $fname=$offs.".exe";
                                open (STORE,">$fname") || die "oops cant write";
                                $storing=$siz;
                    }
        }
        if ($storing) {
                    if ($storing>=512) {
                                syswrite STORE,$data,512;
                                $storing-=512;
                    } else {
                                syswrite STORE,$data,$storing;
                                close STORE;
                                $storing=0;
                    }
        }
        $offs+=512;
}
close IMG;

sub usage {
        print "usage: exetractor.pl <path-and-name-of-dls-image>\n";
        print "the output files will end up in the directory from where\n";
        print "exetractor is being called, so the best way to run it is\n";
        print "cd <emptydir>; exetractor.pl /foo/bar/hd-img.dls\n";
        exit;
}

__END__
```

Some Explanations:
EXE Headers have a particular format that allows to extract the size of
the original file:

```
OFFSET          Description
0000h           2 chars magic number "MZ"
0002h           Number of bytes used in last 512byte block
0004h           Number of 512byte blocks total (including last)
etc
```

Thus, the size of the file can be computed as ([0004h]-1)*512 + [0002h]
This is what the script does with everything that looks like a header of
an EXE file. Then, we naively assume that the original file was nicely
stored in consecutive blocks on the disk. This is of course often not true,
but it works amazingly well :-). If we encounter a new header while we are
still storing what we assume is the previous file, the latter gets aborted
and renamed to ".truncated" to make it obvious.

Files where the number of bytes used in the last 512byte block is indicated
as being bigger than 511 (the block itself) are apparently not proper exes
and will be silently ignored by the script.

```perl
#!/usr/bin/perl
#
# suck.pl – a not very aptly named quick hack to compare a huge list of
# files against a reference of md5sums. Run it with md5sum * | suck.pl
# in the directory where your files are.
#
# ---------------------------------------------------------------------------

while (<>) {
        ($md5sum,$name)=/(\S*)\s*(\S*)/;
        if (exists ($md{$md5sum})) {
                print "Collision: $name and $md{$md5sum}\n";
                next;
        }
        $md{$md5sum}=$name;
        print "Adding $name with $md5sum\n";
}
print "\n\n";
print "Now starting comparison with reference...\n\n"
open (NSRL,"/data/rds/win95/win95_d.md5") || die "oops";
while (<NSRL>) {
        ($md5sum,$name)=/(\S*)\s*(\S*)/;
        if (exists ($md{$md5sum}) ){
                print "$md{$md5sum}\tcould be $name";
                if ($seen{$md5sum}++) {print " (secondary match)"}
                print "\n";
        }
}
close NSRL;
```

Extracting Internet Explorer URLs from a DLS image

The script "iextractor.pl" was written in the course of this analysis and allows to
retrieve Internet Explorer objects from a DLS image, together with the associated
timestamps. An example of the resulting output is shown below. The first number
shown is the access timestamp in hex, to allow for easy sorting. The second item is
the modification time of the object, and the third timestamp the last access time.

```
01c23c02954abea0 Mon Aug  5 00:02:11 2002 Mon Aug  5 00:02:11 2002 Cookie:annemarie@reisen.ch/
01c24322ac782820 Wed Aug 14 01:39:32 2002 Wed Aug 14 01:39:32 2002
Cookie:annemarie@as1.falkag.de/
01c0366dd1764c60 Sun Oct 15 08:04:45 2000 Sun Oct 15 08:04:45 2000 Visited:
annemarie@http://www.altnau.ch
01c24cba426d90e0 Sun Aug 25 15:00:07 2002 Mon Aug 26 06:37:18 2002
http://www.bluewin.ch/bw/image/0,2276,9319,00.jpg
[…]
01c24d7ef006f100 Wed Jul 31 13:07:27 2002 Tue Aug 27 06:05:10 2002
http://www2.bluewin.ch/includes_marcom/images/146x55_sommerhit_d01.gif
01c24d7ef006f100 Mon Aug 26 09:37:08 2002 Tue Aug 27 06:05:10 2002
http://www2.bluewin.ch/includes_marcom/images/146x55_speedfest_d01.gif
01c24d7ef006f100 Thu Aug 22 10:57:31 2002 Tue Aug 27 06:05:10 2002
http://www2.bluewin.ch/includes_shopping/scm/images/magazin_d/italia.gif
01c24d7ef006f100 Thu Aug 22 11:07:40 2002 Tue Aug 27 06:05:10 2002
http://www2.bluewin.ch/includes_shopping/scm/images/magazin_d/beine.gif
01c0b3da9ef6f100 Fri Mar 23 21:48:31 2001 Fri Mar 23 21:48:31 2001 Visited:
annemarie@http://ad.doubleclick.net/noidadi/resultpage.av.monaco.de;kw=vatikanpost;ord=6519647
?
01c0b7c1667e53a0 Wed Mar 28 23:58:04 2001 Wed Mar 28 21:58:04 2001 :2001032820010329:
annemarie@http://www.zfl.uni-bielefeld.de/suchen/suchen.html
01c0b7c0e88359a0 Wed Mar 28 23:54:30 2001 Wed Mar 28 21:54:32 2001 :2001032820010329:
annemarie@:Host: www.bluewin.ch
```

The script used

```perl
#!/usr/bin/perl -w
#
# iextractor.pl is a magic little program to extract Microsoft
# Internet Explorer history, cache and cookie timelines from
# a forensic DLS image. Use this script if you don't have the
# luxury of a complete index.dat file to play with.
#
# Idea and scripting by Daniel Wesemann, July 24, 2004
# You may adapt and use this script as much as you want, but
# please retain the above credits for my clever original idea
# ------------------------------------------------------------

$billytime=116444736; # epoch difference in wintime, centiseconds
$cutoff=631152000; # earlier than 1jan1990 is fishy

while (<>) {
        while (/URL\s.{4}(.{7}[\x01])(.{7}[\x01]).{72}([^\x00]*)/g) {
                $url=$3;
                $a=unpack("x4L",$1); $b=unpack("L",$1);
                next if (($a==0) or ($b==0));
                $modified=&wintime($a,$b);
                next if ($modified < $cutoff);
                $c=unpack("x4L",$2); $d=unpack("L",$2);
                $accessed=&wintime($c,$d);
                $url=~tr/[ -\xaf]/\#/c;
                printf "%08x%08x ",$c,$d; # print hex access time to allow sorting
                print localtime($modified)." ".localtime($accessed)." ".$url."\n";
        }
}

sub wintime {
        my ($a,$b) = @_;
        my $factor=exp(32*log(2)-9*log(10));
        my $aa=$a*$factor;
        my $bb=exp(log($b)-9*log(10));
        my $time=($aa+$bb-$billytime)*100;
        return($time);
}

__END__
```

Some Explanations:

The hardest about reading Internet Explorer stuff is the time format.
Windows stores these times in nanoseconds since 1600, which is very
different from what Unix does AND leads to [expletive] big numbers that
cannot be handled properly on 32bit systems. This is why there is some
odd log/exp stuff in the conversion routine -- all to keep the numbers
from running into overflow. To verify the time conversion routine, feed
the hex 8byte string into "w32tm /ntte" on any Windows box.

Something all entries have in common is the start with "URL<space>",
followed by the modified and accessed date in Wintime format. After-
wards, all bets are off, but URL stuff usually starts 72 bytes further
down the line. It would actually be possible to output cookie contents
(for Cookie entries) and the headers of the webserver response (for
cache entries), but I haven't yet found the patience to code this :-)
Also, for cache entries, modified time seems to mean modification of the
original, not the time when the cache entry was created. But this is just
my guess...

Note that the time values are returned in localtime, so if your evidence
was not created in the same timezone, you are on your own. Also, the time
conversion may be off by one hour, because Windows and Unix seem to
disagree on the concept of daylight saving time. So, no, you cannot
get a conviction with the output of this script :-), but I hope you'll
find it useful nevertheless.

The two scripts "exetractor.pl" and "iextractor.pl" will be made publicly available
through my web page once this paper has been accepted and graded.

Autopsy

The forensic browser "autopsy" was primarily used to manually check on the contents of some disk blocks. Given the structure of the evidence (reformatted), most of the built-in automatic analysis steps of Autopsy did not return anything useful.

Lazarus

While a complete Lazarus analysis was performed against the DLS image, the results did not reveal any important information that had been missed or overlooked during the manual analysis described in the narrative.

References of Documents and Tools used

XVI32   Windows Hex Editor, by Christian Maas
Sleuthkit   forensic tool chest, by Brian Carrier
Perl   The Mother of all Programming Languages, by Larry Wall
Autopsy 2.0   Forensic Browser, by Brian Carrier
VMWare   my virtual lab system, by vmware.com
catdoc   a Microsoft Word "reader" for Linux, by Vitus Wagner
Foremost   Forensic Extraction Tool, by Kris Kendall and Jesse Kornblum
JCDSee   Picture viewer, by Tri Tran, M Neoya and M. Burg

[URL]   What is in index.dat files, www.milincorporated.com (no date given)
[URL]   Reverse Engineering Index.dat, Louis K. Thomas, 2003
[URL]   Time conversion between Win32 and Unix Time, Ian Grigg, 2003
[URL]   Cache management in Internet Explorer 4, Wolfgang Baudisch, 1999
[URL]   Microsoft Word97 File Format, S.R. Haque, 2001
[URL]   Windows95 Alphabetical File Listing, Bob Cerelli (no date given)
[URL]   EXE File Format Header, John Stephenson (no date given)
[URL]   JPG File Format Header, W3.org JPEG Working Group
[URL]   The ID3 Content Description Header, www.id3.org , 2004
[URL]   National Software Reference Library, www.nsrl.nist.gov
[URL]   Virus and Malware information from www.sophos.com