Global Information Assurance Certification Paper

GCFA Practical

Infected or "Owned"?

Anton Chuvakin, Ph.D., GCIA, GCIH

Version 1.5

Submitted: December 1, 2004
Resubmitted: Jan 28, 2005

Abstract

Part I of this practical analyzes the floppy disk obtained from a system administrator at Ballard Industries. The suspicion is that there is a leak of confidential information.

Part II analyzes the disk image taken from a suspected compromised or infected machine. It also suspected that the system administrators have performed certain remediation activities on a compromised machine, before forensics investigators had a chance to take disk image and perform volatile data collection. The latter activities might have destroyed some data related to the infection.

Part I Suspicious Floppy Analysis

Executive Summary

Suspicious floppy was analyzed with computer forensics tools and convincing evidence of proprietary information theft was uncovered.

Analyze an Unknown Image

The following is taken from the assignment with minor abbreviations:

> It was determined that one of Ballard's major competitors, Rift, Inc., has been receiving the new orders for the same fuel cell battery which was once unique to Ballard. A full blown investigation ensues … The only thing out of the ordinary that has turned up is a floppy disk that was being taken **out of the R&D labs** by Robert Leszczynski on 26 April 2004 at approximately 4:45 pm MST, which is against company policy. The **on staff security guard seized the floppy disk** from Robert's briefcase and told Robert he could retrieve it from the security administrator.
>
> Your **primary task** is to analyze this floppy disk and provide a report to Mr Keen. Determine what is on the floppy disk and establish how it might have been used by Mr. Leszczynski.

The above is provided here for reference and for more effective focusing of the investigation.

Examination Details

Describe in detail how you obtained the image and what you did with it after you

received it?

I have collected an image from a security administrator David Keen on a signed (on the paper label) floppy disk. It was handed in person by the administrator. The floppy was labeled with the following:

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

What steps did you take to analyze the image? You should detail out in chronological order the steps you took for your analysis.

I have structured my investigation process in several phases:

I. **Preparation**: at this stage we ready all the tools for an effective and credible forensic investigation that can withstand the court of law. That will be accomplished by carefully documenting all the steps and explaining how they were performed
II. **Analysis**: at this stage we actually study the media and pursue various paths to reach our investigation goals
III. **Reporting**: here we summarize our findings for technical and non-technical audience and formulate our conclusions, tied to the evidence

Preparation went through the following steps listed in chronological order (with documentation to a single log file performed at each step[1]). Each step is concluded with a result (marked 'result') below), that is used in further analysis. Some steps below also define a 'next steps to try', marked with a 'TODO' label:

1. Secure and harden[2] systems to be used for analysis
    a. Main analysis machine: Windows XP SP2 (patched and hardened)
    b. Secondary analysis machine: Linux RedHat 9 (patched and hardened)
    c. Backup machine: Linux RedHat 9 (patched and hardened)
    d. **Result:** secure and predictable systems for the investigation and data storage
2. Make several backup copies of the floppy disk in question to
    a. **Hard disk** of an analysis machine (main working copy)
    HOW: a floppy was inserted into the analysis system and the command
    # dd if=/dev/fd0 of=/home/anton/image-gcfa1.img
    was run. This command copied all the bits from the floppy to the file on the

[1] That is the reason the documentation steps are not present in the overall plan
[2] Machines are secured and hardened according to the industry best practices (patching, limited services, user privileges and network access). Thus secured systems should serve as one of the indications that the stored and analyzed evidence was not tampered with. Detailed discussion of the specifics is clearly out of scope of this document.

analysis machine. 'dd' command has been verified to produce a forensically sounds bit-by-bit copy.

b. **Another floppy** which is then write-protected and labeled (hard backup copy 1)

HOW: a blank new floppy was inserted into the analysis machine and the command

 # dd if=/home/anton/image-gcfa1.img of=/dev/fd0

This created a copy of the original floppy. The correctness of the copies will be verified in the next step.

c. **Hard disk** of a backup machine (backup copy 2)

HOW: the forensic image was copied from the analysis machine to the backup machine via SSH. The command

$ scp /home/anton/image-gcfa1.img anton@abackup:~/GCFA-back/

was run. It copies the file (bit by bit copy that is forensically sound) from one machine to another in a secure (i.e. encrypted and authenticated) manner

d. Result: a set of working and backup copies of the original evidence

3. Checksumming the image
   a. Compute the MD5 sums of the images (working, backup)

   HOW: we computed the md5 checksums of all the floppy images. The commands

   $ md5sum image-gcfa1.img

   $ md5sum /dev/fd0

   were used to compute the checksums of the file (on the analysis and backup machines) and the duplicate floppy image respectively

   b. Compare with the one provided by the admin on a floppy label

   HOW: the md5 strings obtained from the above commands were visually compared to the one present on the original floppy label

   c. Result: all checksums match
4. Chain of custody measures
   a. Protect the original diskette in a safe

   HOW: diskette was stored in the company safe with keys only accessible to CFO and his stuff of 3.

   b. Protect the backup diskette in a different safe

   HOW: the duplicate diskette was stored in a safe owned by the consulting company running the investigation. Only 2 people knew the combination.

   c. **Result**: original is protected from the elements and human errors
5. Prepare the tools available in the forensic toolkit
   a. AccessData FTK (main analysis tool)

   HOW: AccessData FTK was installed from an official CD personally obtained by the investigator from the company sales representative.

      i. The software was installed on the Windows analysis machine by running the 'setup.exe' command and then following prompts.

      ii. As a result, the tool is installed and can be launched by clicking an FTK icon on a Windows Desktop.

   b. Sleuthkit/Autopsy (secondary analysis tool)

HOW: the software was downloaded to the Linux analysis machine from the site www.sleuthkit.org via a *wget* command (a standard Linux command to download files):

i. $ wget **http://internap.dl.sourceforge.net/sourceforge/sleuthkit/sleuthkit-1.73.tar.gz**

ii. wget **http://internap.dl.sourceforge.net/sourceforge/autopsy/autopsy-2.03.tar.gz**

iii. verified with a checksum on the websites: sleuthkit md5 is 773c48dd05caa0262d72015498fd92ce[3] and autopsy md5 is 51b056624cc81ca1bdf281e2e23a160d[4]. The checksum was verified by running commands:

*$ md5sum **autopsy-2.03.tar.gz***

*$ md5sum **sleuthkit-1.73.tar.gz***

iv. and then installed like this:

$ tar zxf sleuthkit-1.73.tar.gz

$ cd sleuthkit-1.73

$ make

the commands above unpack the tool source archive and build the *sleuthkit* forensic tool in the appropriate directory

$ tar zxf autopsy-2.03.tar.gz

$ make

After one follows the directions of the *make* script, the *autopsy* front-end for *sleuthkit* is installed. It can now be launched by typing:

$ ./autopsy

and pointing a web browser on the same system to the address shown by the autopsy upon the launch.

c. Standard UNIX tools listed below

HOW: the authenticity of tools installed on a Linux system was verified via the *rpm* command that computes a cryptographic checksum of each file within a software package and compares it to the "known good" database, installed on the same system from the installation CD:

$ rpm –V binutils

$ rpm – V fileutils

$ rpm –V tar

$ rpm –V wget

the above commands verified packages containing all the needed tools referenced below (such as *tar, cp, strings, dd*, etc) and should show **no** output when executed.

Result: "known good" tools prepared for the investigation

6. Load the floppy disk image into the forensics tool

HOW: here is how FTK was used to load the image and prepare for the investigation

1. launch FTK

[3] Obtained from http://www.sleuthkit.org/sleuthkit/download.php

[4] Obtained from http://www.sleuthkit.org/autopsy/download.php

2. choose ‘New Case’ when prompted
3. choose ‘Add Evidence’ when prompted or go to a ‘File’ menu and then choose ‘Add Evidence’

HOW: here is how Sleuthkit/Autopsy was used to load the image for investigation

i. Launch autopsy as shown above
ii. Choose open case
iii. Choose the host and click on ‘OK’
iv. Choose the image and click on ‘OK’
v. Follow further instructions to follow into analysis mode

7. Both tools display the md5 sums of the images so that the investigator can compare them to “known correct” ones.

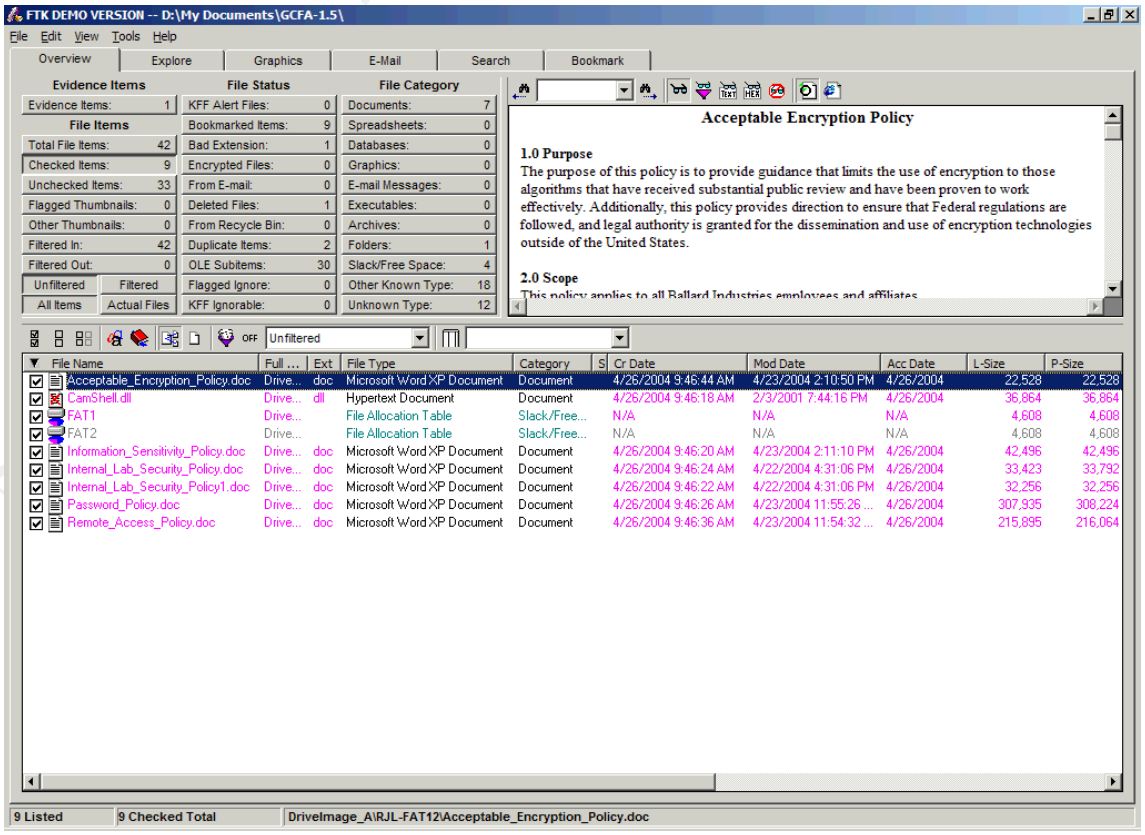
That concluded the preparation stage. At this stage we are ready to proceed with the analysis, knowing that all the evidence is safe and secure and can not be studied.

Analysis starts with loading the image into the appropriate products.

a. Preliminary analysis

1. File inspection

HOW: both forensic tools used show files present on the floppy. A standard case view in Autopsy and FTK shows such files and no special action is required on the user behalf to see that. For example, this screen show shows all the files present on the image:



2. Deleted object inspection

HOW: both forensic tools used show deleted files present on the floppy. A standard case view in Autopsy and FTK shows such files and no special action is required on the user behalf to see that. The above screen shot shows the deleted files. Their contents (where accessible on disk and not overwritten) can be views by clicking on a file.

3. Empty (slack, unallocated, etc) space inspection

HOW: FTK allows for easy inspection of the unallocated space. Here is how to do this:

- Open the FTK and then open the appropriate case, evidence will be shown
- Click the button 'Slack/Free Space', the viewer pane will show the relevant items which can be browsed using the top right view panel (see above screenshot for a similar view)

4. **Result:** identified current and deleted files

b. Export files and deleted files to the working directory

HOW: both tools allow extracting files from images for inspection. Here is how to do it in FTK:

- Launch the tool and open the case
- Click on one of the button referring to the files, such as 'Total File Items', the file selections will appear in the bottom windows panel
- Right click on the file that needs to be exported or select multiple files and then right-click
- Choose export from the menu that appears
- Follow further directions to export to the desired location on the disk

Result: files from the image are copied to the analysis system

c. Focus on files

1. Review extracted files using the associated application (MS Word in this case) and look for suspicious signs

HOW: after we extracted the files we opened them using the associated application (MS Word 2003) and looked at their content. This was accomplished by clicking on file names on the analysis system. We did not review the recovered deleted file with type '.dll' and focused on the documents

Result: boring files with corporate policy data extracted

2. **TODO**: need detailed review in hex editor and metadata review for next level analysis

HOW: Using the hex editor (standalone or built-in into the forensics tools) one can review the actual content of the files and/or the printable characters ('strings') within the file. To extract the printable characters we run:

$ strings CamShell.dll > CamShell.dll.strings

on our Linux analysis machine. This creates a file *CamShell.dll.strings* containing all the printable characters within a file (file contents are shown below).

We will also use the editor built-in into FTK. To view the file contents

the user needs to click on a file and the contents will appear in the upper right panel. The desired mode (HEX – showing the hexadecimal representation of all the file content) the button needs to be pressed.

d. Focus on deleted content

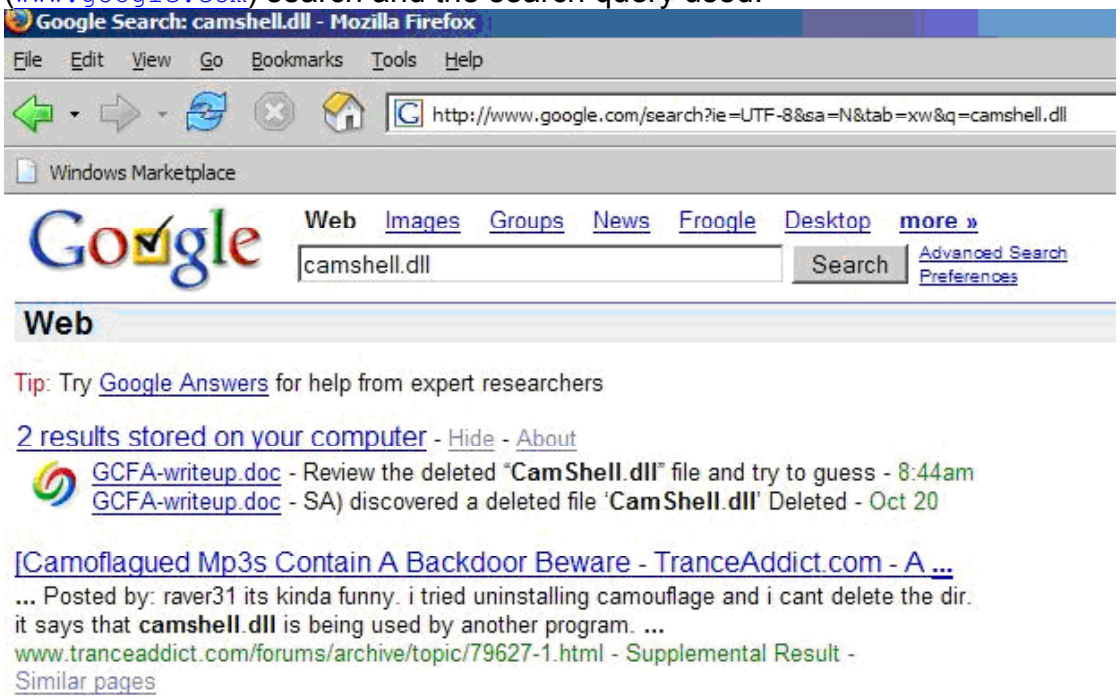
1. Review the deleted "CamShell.dll" file and try to guess its purpose

HOW: we look at the file contents in the FTK and also review the strings file produced the above by opening the strings file in the file viewer:

$ less CamShell.dll.strings

2. Google for "camshell.dll"

HOW: see Figure below for the results of the Google (www.google.com) search and the search query used.



Result: discover that the name of the file coincides with the name of the component of the known data hiding program *Camouflage* (!)

3. **TODO**: detailed analysis of the program needed. However, a dll file by itself cannot be run and we can only review its contents (rather than run it).

e. Focus on "empty" space

1. Scroll through unallocated space looking for things of interest[5]

HOW: we click on the "Slack/Free Space" button in the FTK and then manually review the unallocated space.

For verification purposes we also use a tool "dls" (a part of Sleuthkit)

[5] This is possible since we have the small amount of disk space (one floppy); for bigger investigation this step is likely impossible

that extracts the unallocated space from an image file into a file:
$ dls –f fat16 myimage.img > myimage-deleted.img
The command uses the '-f' flag for a file system type (a standard Windows FAT) and will extract the space into a file. The results can now be reviewed either manually or from within the *Autopsy* front-end by clicking on the 'All deleted files'

2. **Result**: we discover that at the very beginning of a floppy image a chunk of HTML data (same type that is used in web pages) is found. Its significance for the case has not been established. The HTML appears to overwrite the beginning of the CamShell.dll file. The rest of the free space contains zeros which is appropriate for a new or "wiped"[6] floppy

f. Focus on timeline

1. Use the forensics tools to restore the incident timeline (Figure below)

HOW: we generate a time lines using Autopsy by clicking "File Activity Timelines" after opening the case and following to host and image. One needs to prepare a data file suitable for timelines analysis for an image file (click 'Create Data File' first). Next, one has to create an actual timeline from the above file (click 'Create Timeline' next). After this, one can view or export the timeline to a text file.

We can also generate a timeline manually by running a tool 'mactime' included in the Sleuthkit. First, one needs to prepare a data file.

a. $ fls –m /mnt/floppy –t fat12 read-v1_5 > read-v1_5.body
this command extracts file timestamps into a machine-readable ASCII format as if the disk was mounted as /mnt/floppy (flag '-m') and using Windows standard floppy file systems type (flag '–t fat12'). The file contains lines with file names, sizes, permissions and timestamps.
This file looks like this:

```
0|/mnt/floppy/RJL        (Volume Label Entry)|0|3|33279|-/-
rwxrwxrwx|1|0|0|0|0|1082865600|1082904820|1082904820|512|0
0|/mnt/floppy/CamShell.dll (_AMSHELL.DLL) (deleted)|0|5|33279|-/-
rwxrwxrwx|0|0|0|0|36864|1082952000|981247456|1082987178|512|0
0|/mnt/floppy/Information_Sensitivity_Policy.doc (INFORM~1.DOC)|0|9|33279|-/-
rwxrwxrwx|1|0|0|0|42496|1082952000|1082743870|1082987180|512|0
0|/mnt/floppy/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)|0|13|33279|-/-
rwxrwxrwx|1|0|0|0|32256|1082952000|1082665866|1082987182|512|0
0|/mnt/floppy/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)|0|17|33279|-/-
rwxrwxrwx|1|0|0|0|33423|1082952000|1082665866|1082987184|512|0
0|/mnt/floppy/Password_Policy.doc (PASSWO~1.DOC)|0|20|33279|-/-
rwxrwxrwx|1|0|0|0|307935|1082952000|1082735726|1082987186|512|0
0|/mnt/floppy/Remote_Access_Policy.doc (REMOTE~1.DOC)|0|23|33279|-/-
rwxrwxrwx|1|0|0|0|215895|1082952000|1082735672|1082987196|512|0
```

[6] Wipe = delete securely making sure that the data is overwritten on the disk. It is sometimes accomplished by writing zeros to the disk, producing the result similar to the observed in this case.

```
0|/mnt/floppy/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)|0|27|33279|-/-
rwxrwxrwx|1|0|0|0|22528|1082952000|1082743850|1082987204|512|0
0|/mnt/floppy/_ndex.htm (deleted)|0|28|33279|-/-
rwxrwxrwx|0|0|0|0|727|1082952000|1082732036|1082987256|512|0
```

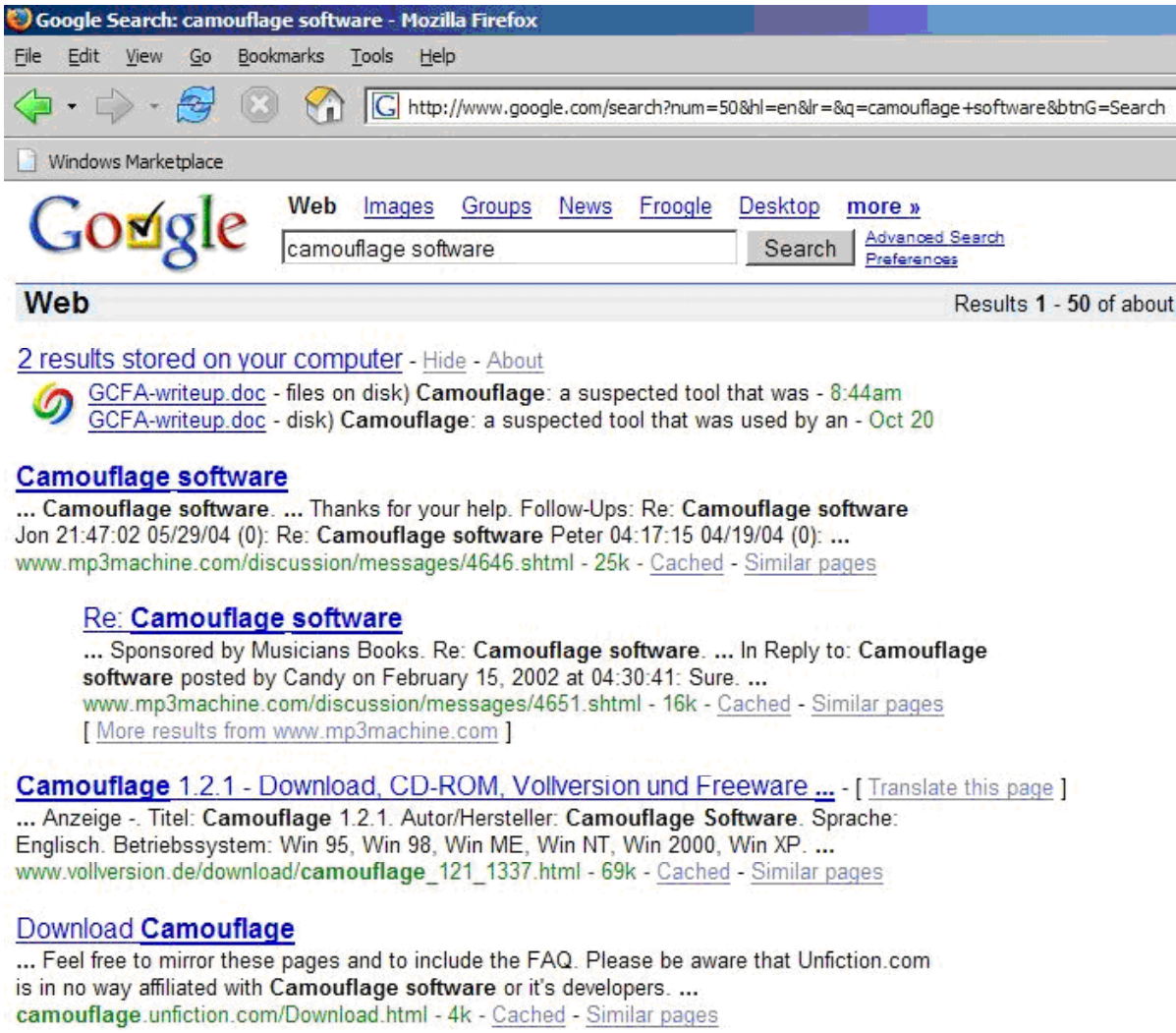
b. $ mactime –b read-v1_5.body > read-v1_5.timeline
This command creates a human readable timeline from a body machine readable file. The resulting file looks like this (excerpt):

```
Sat Feb 03 2001 19:44:16    36864 m.. -/-rwxrwxrwx 0        0        5        /mnt/floppy/CamShell.dll
(_AMSHELL.DLL) (deleted)
Thu Apr 22 2004 16:31:06    33423 m.. -/-rwxrwxrwx 0        0        17
/mnt/floppy/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
                            32256 m.. -/-rwxrwxrwx 0        0        13
/mnt/floppy/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Fri Apr 23 2004 10:53:56      727 m.. -/-rwxrwxrwx 0        0        28       /mnt/floppy/_ndex.htm (deleted)
Fri Apr 23 2004 11:54:32   215895 m.. -/-rwxrwxrwx 0        0        23
/mnt/floppy/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26   307935 m.. -/-rwxrwxrwx 0        0        20       /mnt/floppy/Password_Policy.doc
(PASSWO~1.DOC)
```

A full timeline for this case is shows below.

g. Perform detailed analysis of the CamShell.dll file
    1. More Google searches help identify the purpose and functionality of a data hiding program

HOW: see Figure to learn how we searched more in Google.



2. Program web site discovered

HOW: we found a site that distributes this program. This site is http://camouflage.unfiction.com/Download.html

To conclude that it is indeed the same program, we need to install it on the analysis system.

3. Software downloaded and installed on the analysis machine

HOW: file is downloaded from http://camouflage.unfiction.com/Camou121.exe , saved on the analysis machine as *Camou121.exe* and then installed by clicking on the file. Several files are installed in *C:\Program Files\Camouflage* as a result

4. Component files are identified (Figure 4)

HOW: we look in the *C:\Program Files\Camouflage* and see the new files

5. Recovered dll file is compared[7] with a dll file from the installed package

[7] File comparison is performed at 3 levels: size, MD5, manual review

HOW: we look at the file using several means.
- First, we try to run a 'file' command on the recovered dll file. 'File' shows us the file type (such as document, executable, etc):
$ file CamShell.dll
the result is "HTML document text", which is puzzling at that stage.
- We now look at the file contents and see that the beginning of it seems dissimilar from other dll files. It fact, it looks like HTML.
- We conclude that the file was overwritten after deletion and thus md5 comparison will not show similarity
- Now we manually review both the recovered file and the file installed by Camouflage in hex editor/viewer to see whether the body of the file is the same *after* the initial part overwritten by the HTML
- We observe that the file contents appear to be the same starting from the end of the HTML code
- Thus, we can strengthen our program identification, but still cannot say it convincingly due to program contents overwrite and lack of other installed Camouflage components

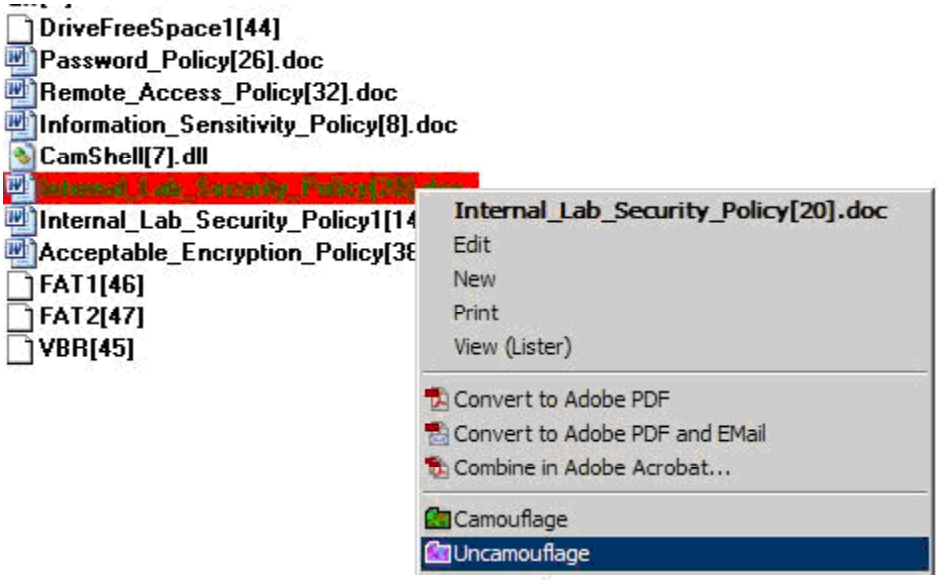
6. **Result**: data hiding program identified

h. Using the Camouflage program for data recovery

1. Utilize the installed Camouflage program on doc files recovered from image

HOW:
- Upon installation, new options *Camouflage/Uncamouflage* become available in the Windows Explorer right click menu
- After clicking on one of the recovered files we choose *Uncamouflage*
- The program asks for a password and we do not enter any (just press 'Enter' on the keyboard), hoping that the offender did not go to that level of extra security
- if the file is extracted the program indicates that and saves the file on disk

For example:



2. All recovered word files are tested and one ("Internal_Lab_Security_Policy.doc"[8]) is successfully "uncamouflaged" without a password (!)
3. **Result**: hidden message discovered (Figure below)

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

i. Attempts further recovery based on the above message
   1. Based on the sizes and structure of other files, we hypothesize that they also contain hidden messages (they files are larger that files containing the same text and no other extraneous payload[9])
   2. Now that the other files did not uncamouflage without a password we try to use the hint from the message about the "First Name" (see Appendix A for tried passwords)
   3. **Result**: password guessing unsuccessful, however, other files are likely camouflaged since they contain random-looking content that is not present if visible text is copied to another Word document (experiments revealed that file sized are significantly different between the recovered originals and files created by the investigator by copying the visible content)
   4. Now we can attempt to see if the camouflage is faulty and the

[8] It appears that the file "Internal_Lab_Security_Policy.doc" is its original uncamouflaged version, since the difference in size is close to the size of the note (even considering encryption and possible compression)
[9] Tested by creating the files with the same content in MS Word, the resulting size is much smaller

program persists the unencrypted password in the file itself

- We create our own file Word file “Test.doc” and camouflage a text file “Secret.txt’ within it with a password ‘WeirdTest’
- We run strings on a resulting increased ‘Test.doc’ that now contains the secret message

$ strings Test.doc > Test.doc.str

- We now review the strings hoping to find the password or some weakly encrypted version of it

$ grep –i weirdtest Test.doc.str

the above command utilizes Linux *grep* utility with ‘-i’ parameter (for case insensitive matching) to look for a string in a file. We use case insensitivity in case the passwords are not case sensitive and can appear in either small or capital letters[10].

- We do not find the password in clear text anywhere in the file, but possibility still remains that a weak encryption scheme is in use

5. We can also hope to brute force the password out of the program (in case). This can be accomplished by trying various password combinations or random strings. This process is likely to take a long time and was not performed for this investigation.

j. Further file recovery research was performed on Google

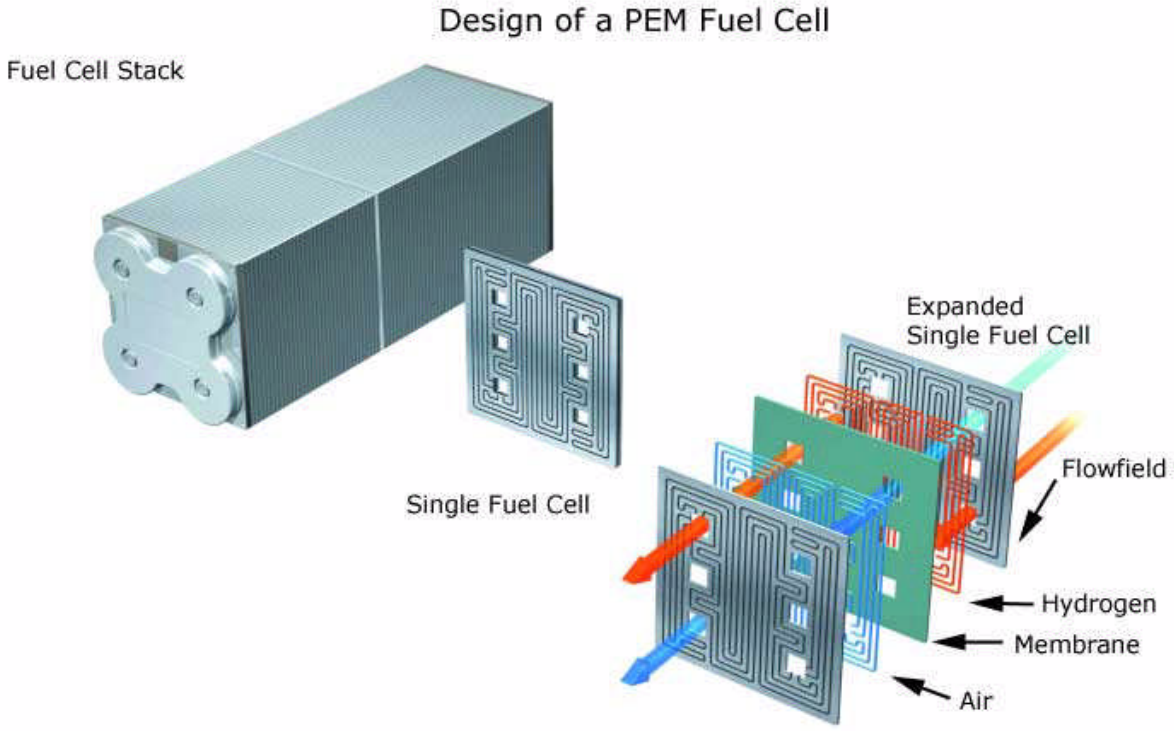
1. we search for “camouflage software password recovery”; the third link is “(easily) Breaking a (very weak) steganography software: Camouflage” (http://www.guillermito2.net/stegano/camouflage/) which appears to be a research paper on breaking Camouflage “protection”. It appears that the password can be recovered from a file by XORing a fixed string with a specific string from a certain location in a file[11]. Such trivial obfuscation is commonly use for an extremely “light” password protection. This was discovered by analyzing files camouflaged with various known passwords (including blank passwords) and comparing the resulting camouflaged files. The author of the research has created a small utility that can be used to extract the password from the camouflaged files.
2. We download the utility “Camouflage_Password_Finder_02.zip”, uncompress it and run the file “Camouflage_Password_Finder.exe”, contained within the Zip archive
3. The utility asks for a name of a camouflaged file, and we try giving all the names of our potentially camouflaged files. Password

[10] We later learned that Camouflage passwords are case-sensitive

[11] "The conclusion is that the password is stored at this position, probably masked by XORing it with a key composed by a fixed string of bytes. This string is now easy to obtain. Because XOR is reversible, you just have to XOR the above data with the password, which is "aaaa...", so in hexadecimal "61616161...” – quoted from “(easily) Breaking a (very weak) steganography software: Camouflage” (http://www.guillermito2.net/stegano/camouflage/)

recovery is successful on 2 of the files recovered from the diskette. The file “Remote_Access_Policy.doc” seems to have a password of “Remote” and the file “Password_Policy.doc” seems to have a password of “Password” (which is a *first name* of the file!)

4. We use our copy of the Camouflage software to uncamouflage the file contents using the recovered passwords. “Password_Policy.doc” contains 3 JPG images with cell schematics and research paper excerpts; the file “Remote_Access_Policy.doc” contains a MS Access database file “CAT.mdb” with what looks like a customer database. One of the images is shown here:



5. We open the recovered JPG and MDB files using the associated applications (viewer and MS Access). The Access database file contains a list of people with their company affiliation and contact information.

Result: we discover mode hidden content which will likely make the case of intellectual property theft even stronger, provided that the files will be verified as authentic by the Ballard representatives.

Reporting stage is described below. The main result is that the person in question has most likely tried and successfully sneaked the data out.

Preliminary Analysis Results

1. Several files were found on a floppy (all of the MSWord document type)
2. FTK and Sleuthkit/Autopsy (SA) discovered a deleted file '*CamShell.dll*'
3. Deleted file is corrupted due to content overwriting but we are still able to identify it is a data hiding program with a high degree of certainty
4. Documents we discovered to contain a hidden message and data files (images and a database) indicating a likely case of intellectual property theft (but verification is still needed)

What tools did you use?

- *AccessData FTK* v. 1.50: commercial forensics tool from AccessData (www.accessdata.com) - demo version limited to 5000 file entries (obtained from AccessData representative at a trade show)
- *Sleuthkit/Autopsy*: open source forensics tool from Brian Carrier (obtained from www.sleuthkit.org)
- Standard UNIX tools: *grep* (pattern matching), *find* (locate files on disk), *strings* (extract printable characters from files), etc
- *Camouflage*: a suspected tool that was used by an attacker (obtained from http://camouflage.unfiction.com/Download.html)
- Google: best general purpose search engine to find references to tools and background material (www.google.com)
- *Camouflage_Password_Finder*: a utility to break the "encryption" of the Camouflage and recover the password from the file (http://www.guillermito2.net/stegano/camouflage/)

1. Explain what Mr. Leszczynski tried to accomplish and if he was Successful.

He apparently tried to sell some company confidential information to a competitor. We can conclude that from the note we discovered in one of the camouflaged files, as well from the presence of hidden data within other camouflaged files and the camouflage software on the floppy.

After we verify the authenticity of schematics and the database with Ballard, we can conclude that he has gained unauthorized access to such files and attempted to carry them out of the facility.

We cannot be 100% sure that he was successful in selling the information, however. Indeed, we certainly know that:

a. There was a leak of confidential information
b. Mr. Leszczynski was caught in possession of a data hiding tool
c. Mr. Leszczynski was caught in possession of a document (i.e. his note shown

above) stating that he possesses and plans to sell the data similar to what was leaked recently
d. Mr. Leszczynski was caught in possession of confidential schematics and a customer database (pending verificvation)

but a. , b., c. and d. together do not make a conviction that he *has sold* the data. Still, it is likely that he indeed was successful, but we cannot make a positive conclusion about that.

2. What did he try to do.

He allegedly tried to:

a. get out an information about the company latest schematics
b. get out a copy of a customer database
c. sell a. and b. to a competitor for $5m

He tried to do that by hiding the files on a floppy and then trying to carry the floppy from the organization facility.

3. What if any information was released?

If information stated in his note was indeed release, Ballard Industries can suffer a significant financial damage.

Leak of customer database can lead to:

1. Targeted solicitation of business by a competitor
2. Targeted marketing campaign by a competitor
3. Breach of privacy lawsuit by a customer

Leak of latest schematics can lead to:

4. Boost to development efforts by competitor
5. Better targeted competitive analysis by a competitor
6. Targeted marketing campaign by a competitor

4. What advice can you provide to the Systems Administrators to help them detect whether there systems have been tampered by Mr. Leszczynski?

While it is possible that Mr Leszczynski has used his *authorized* access to obtain a copy of confidential information referenced above (on the other hand, he could have been using stolen access credential, which will exacerbate his guilt), he was caught using unauthorized software[12] to conceal it. Thus, it is conceivable that he have used hacker

software to break into systems and/or secure his ability to access them (backdoor, etc). In light of the above, the following steps are defined for system administrators and IT managers to follow:

It is **required** to:

- Disable all computer accounts belonging to Mr Leszczynski
- Suspend his employment until the investigation is complete
- Remove his office access privileges
- Conduct a full investigation of the machines that housed the confidential information
- Conduct a full investigation of other machines touched by Mr Leszczynski
- Determine the specific route that information was accessed and leaked out
- Investigate who else had access to confidential information and review their access control privileges

It is **recommended** to:

- Change other passwords within the company
- Review access privileges and apply necessary privilege tightening
- Educate employees on confidentiality of corporate information assets and other security awareness issues

It is *suggested* to:

- Audit other IT resources for possible policy violations and information leaks
- Review and optimize incident management and investigation policies within the company

Image Details:

Listing of all the files in the image.

Here is the list in text format:

Intact:

1. Acceptable_Encryption_Policy.doc
2. Information_Sensitivity_Policy.doc
3. Internal_Lab_Security_Policy1.doc
4. Internal_Lab_Security_Policy.doc
5. Password_Policy.doc
6. Remote_Access_Policy.doc

Deleted:

[12] It will definitely help to confirm that this software was indeed unauthorized and not part of standard company setup (very unlikely but still possible)

1. CamShell.dll
2. *INDEX.html*[13] - partially overlaps with CamShell.dll (2 shared disk clusters)

This was obtained by the forensic tools used as described above.

True name of the program/file used by Mr. Leszczynski.

Camouflage v. 1 by Twisted Pear Productions, www.camouflagesoftware.com (dead) or http://camouflage.unfiction.com/Download.html (active 11/15/2004)

File/MACTime information from image(last modified, last accessed and last changed time).

We will use multiple forensics tools to achieve higher certainty of information. We will investigate all tool conflicts separately!

To obtain file/MACTime information in FTK, one needs to:
- Open the image, MAC times will be shows to the user

File Name	Full ...	Ext	File Type	Category	Cr Date	Mod Date	Acc Date	L-Size	P-Size
Remote_Access_Policy.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:36 AM	4/23/2004 11:54:32 ...	4/26/2004	215,895	216,064
Password_Policy.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:26 AM	4/23/2004 11:55:26 ...	4/26/2004	307,935	308,224
Internal_Lab_Security_Policy1.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:22 AM	4/22/2004 4:31:06 PM	4/26/2004	32,256	32,256
Internal_Lab_Security_Policy.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:24 AM	4/22/2004 4:31:06 PM	4/26/2004	33,423	33,792
Information_Sensitivity_Policy.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:20 AM	4/23/2004 2:11:10 PM	4/26/2004	42,496	42,496
CamShell.dll	Drive...	dll	Hypertext Document	Document	4/26/2004 9:46:18 AM	2/3/2001 7:44:16 PM	4/26/2004	36,864	36,864
Acceptable_Encryption_Policy.doc	Drive...	doc	Microsoft Word XP Document	Document	4/26/2004 9:46:44 AM	4/23/2004 2:10:50 PM	4/26/2004	22,528	22,528
[Root Folder]	Drive...		Root Folder	Folder	N/A	N/A	N/A	7,168	7,168

To obtain file/MACTime information in Sleuthkit/Autopsy, one needs to:
- Open the image, MAC times will be shows to the user

[13] According to 1 out of 2 forensics tools: FTK did not see this file, while Autopsy/Sleuthkit did see it

Del	Type dir / in	Name	Written	Accessed	Created	Size
✓	r / r	_ndex.htm	2004.04.23 10:53:56 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:47:36 (EDT)	727
	r / r	Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)	2004.04.23 14:10:50 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:44 (EDT)	22528
✓	r / r	CamShell.dll (_AMSHELL.DLL)	2001.02.03 19:44:16 (EST)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:18 (EDT)	36864
	r / r	Information_Sensitivity_Policy.doc (INFORM~1.DOC)	2004.04.23 14:11:10 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:20 (EDT)	42496
	r / r	Internal_Lab_Security_Policy.doc (INTERN~2.DOC)	2004.04.22 16:31:06 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:24 (EDT)	33423
	r / r	Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)	2004.04.22 16:31:06 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:22 (EDT)	32256
	r / r	Password_Policy.doc (PASSWO~1.DOC)	2004.04.23 11:55:26 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:26 (EDT)	307935
	r / r	Remote_Access_Policy.doc (REMOTE~1.DOC)	2004.04.23 11:54:32 (EDT)	2004.04.26 00:00:00 (EDT)	2004.04.26 09:46:36 (EDT)	215895
	r / r	RJL (Volume Label Entry)	2004.04.25 10:53:40 (EDT)	2004.04.25 00:00:00 (EDT)	2004.04.25 10:53:40 (EDT)	0

The data in a table below was correlated from the above tools. Keep in mind, that FAT file system (FAT12 is used on a floppy) has severe limitation on timestamps recorded (e.g. see http://www.microsoft.com/resources/documentation/Windows/XP/all/reskit/en-us/Default.asp?url=/resources/documentation/windows/xp/all/reskit/en-us/prkc_fil_yksz.asp). Namely, it does not record the precise timestamp of the time when the file was last accessed.

File	Created	Last Modified	Last Accessed	Last Changed[14]
Acceptable_Encryption_Policy.doc	4/26/2004 9:46:44 AM	4/23/2004 2:10:50 PM	4/26/2004[15]	N/A
Information_Sensitivity_Policy.doc	4/26/2004 9:46:20 AM	4/23/2004 2:11:10 PM	4/26/2004	N/A
Internal_Lab_Security_Policy1.doc	4/26/2004 9:46:22 AM	4/22/2004 4:31:06 PM	4/26/2004	N/A
Internal_Lab_Security_Policy.doc	4/26/2004 9:46:24 AM	4/22/2004 4:31:06 PM	4/26/2004	N/A
Password_Policy.doc	4/26/2004 9:46:26 AM	4/23/2004 11:55:26 AM	4/26/2004	N/A
Remote_Access_Policy.doc	4/26/2004 9:46:36 AM	4/23/2004 11:54:32 AM	4/26/2004	N/A
CamShell.dll	4/26/2004 9:46:18 AM	2/3/2001 7:44:16 PM	4/26/2004	N/A

[14] Last change timestamp is not present on FAT filesystems
[15] Not accurate on FAT file systems (only contains a date field, but no time)
[5] According to 1 out of 2 forensics tools: FTK did not see this file, while Autopsy/Sleuthkit did see it

INDEX.HTM[5]	4/26/2004 9:47:00 AM	4/23/2004 10:53:00 AM	4/26/2004	N/A

There is a peculiarity between the **Created** and **Last Modified** timestamps. However, it is explained by the fact that the files the Creation time stamp refers to the time that files were copied to the floppy (i.e. created in its root directory), while the Modification stamp refers to their modification on the other machine (likely his workstation). See references for the information about timestamp persistence across file operations (e.g. http://www.jsiinc.com/SUBI/tip4100/rh4154.htm)

Here is the timeline analysis as generated by the Sleuthkit/Autopsy:

TIME	SIZE	TYPE	PERMISSIONS			CLU[17]	FILE NAME
Thu Apr 22 2004 16:31:06	32256	m..	-/-rwxrwxrwx	0	0	13	a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
	33423	m..	-/-rwxrwxrwx	0	0	17	a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Fri Apr 23 2004 10:53:56	727	m..	-/-rwxrwxrwx	0	0	28	a:\/_ndex.htm (deleted)
	727	m..	-rwxrwxrwx	0	0	28	<read-v1_5-_ndex.htm-dead-28 >
Fri Apr 23 2004 11:54:32	215895	m..	-/-rwxrwxrwx	0	0	23	a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Fri Apr 23 2004 11:55:26	307935	m..	-/-rwxrwxrwx	0	0	20	a:\/Password_Policy.doc (PASSWO~1.DOC)
Fri Apr 23 2004 14:10:50	22528	m..	-/-rwxrwxrwx	0	0	27	a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Fri Apr 23 2004 14:11:10	42496	m..	-/-rwxrwxrwx	0	0	9	a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Sun Apr 25 2004 00:00:00	0	.a.	-/-rwxrwxrwx	0	0	3	a:\/RJL (Volume Label Entry)
Sun Apr 25 2004 10:53:40	0	m.c	-/-rwxrwxrwx	0	0	3	a:\/RJL (Volume Label Entry)
Mon Apr 26 2004 00:00:00	215895	.a.	-/-rwxrwxrwx	0	0	23	a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
	307935	.a.	-/-rwxrwxrwx	0	0	20	a:\/Password_Policy.doc (PASSWO~1.DOC)
	42496	.a.	-/-rwxrwxrwx	0	0	9	a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
	727	.a.	-/-rwxrwxrwx	0	0	28	a:\/_ndex.htm (deleted)
	32256	.a.	-/-rwxrwxrwx	0	0	13	a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
	727	.a.	-rwxrwxrwx	0	0	28	<read-v1_5-_ndex.htm-dead-28 >
	33423	.a.	-/-rwxrwxrwx	0	0	17	a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
	36864	.a.	-/-rwxrwxrwx	0	0	5	a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
	36864	.a.	-rwxrwxrwx	0	0	5	<read-v1_5-_AMSHELL.DLL-dead-5 >

[5] According to 1 out of 2 forensics tools: FTK did not see this file, while Autopsy/Sleuthkit did see it
[17] CLU stands for file system clusters

	22528	.a.	-/-rwxrwxrwx	0	0	27	a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:46:18	36864	..c	-/-rwxrwxrwx	0	0	5	a:\/CamShell.dll (_AMSHELL.DLL) (deleted)
	36864	..c	-rwxrwxrwx	0	0	5	<read-v1_5-_AMSHELL.DLL-dead-5 >
Mon Apr 26 2004 09:46:20	42496	..c	-/-rwxrwxrwx	0	0	9	a:\/Information_Sensitivity_Policy.doc (INFORM~1.DOC)
Mon Apr 26 2004 09:46:22	32256	..c	-/-rwxrwxrwx	0	0	13	a:\/Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
Mon Apr 26 2004 09:46:24	33423	..c	-/-rwxrwxrwx	0	0	17	a:\/Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
Mon Apr 26 2004 09:46:26	307935	..c	-/-rwxrwxrwx	0	0	20	a:\/Password_Policy.doc (PASSWO~1.DOC)
Mon Apr 26 2004 09:46:36	215895	..c	-/-rwxrwxrwx	0	0	23	a:\/Remote_Access_Policy.doc (REMOTE~1.DOC)
Mon Apr 26 2004 09:46:44	22528	..c	-/-rwxrwxrwx	0	0	27	a:\/Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
Mon Apr 26 2004 09:47:36	727	..c	-rwxrwxrwx	0	0	28	<read-v1_5-_ndex.htm-dead-28 >
	727	..c	-/-rwxrwxrwx	0	0	28	a:\/_ndex.htm (deleted)

Where:

- CLU stands for clusters
- In TYPE column: ‘a’ is for access, ‘m’ is for modification, ‘c’ is for creation (i.e. copying or moving to floppy)

To obtain the above timeline, we select the ‘File Activity Timelines’ option in the Autopsy menu, generate the timeline file from the image and the display or export it (see description earlier in the document). The above is a timeline display within the Autopsy interface.

File owner(s) (user and/or group).

This is a floppy with a FAT12 file system. There is no file owner (UID/GID) in FAT.

File size (in bytes).

See the table below. Information is correlated from multiple tools, same as for the timeline above

File	Size[18], bytes
Acceptable_Encryption_Policy.doc	22528
Information_Sensitivity_Policy.doc	42496

[18] Excluding slack size (as reported by FTK)

Internal_Lab_Security_Policy1.doc	32256
Internal_Lab_Security_Policy.doc	33423
Password_Policy.doc	307935
Remote_Access_Policy.doc	215895
CamShell.dll	36864
INDEX.HTM[5]	727

File sized in the above table to not count possible slack space (i.e. free disk space in the very last occupied cluster, not taken by the file data). Some forensic tools, such as FTK used here report both sizes: real (also called "logical") and on-disk size (with the slack space up to the end of the last occupied cluster).

Both tools (FTK and Autopsy) show file sizes in the main view next to names and no user action is needed to view the sizes.

MD5 hash of the file (include screen shots of the hash value obtained).

See the table below. Information is correlated from multiple tools; same as for the timeline above (see screenshots above for the original md5sums as displayed by the tools)

File	MD5SUM
Acceptable_Encryption_Policy.doc	f785ba1d99888e68f45dabeddb0b4541
Information_Sensitivity_Policy.doc	99c5dec518b142bd945e8d7d2fad2004
Internal_Lab_Security_Policy1.doc	e0c43ef38884662f5f27d93098e1c607
Internal_Lab_Security_Policy.doc	b9387272b11aea86b60a487fbdc1b336
Password_Policy.doc	ac34c6177ebdcaf4adc41f0e181be1bc
Remote_Access_Policy.doc	5b38d1ac1f94285db2d2246d28fd07e8
CamShell.dll	6462fb3acca0301e52fc4ffa4ea5eff8
INDEX.HTM[5]	17282ea308940c530a86d07215473c79

To obtain md5 information in FTK one needs to:

1. Open the evidence image
2. Right-click on a file , a menu will appear
3. Choose 'File Properties', a windows will open
4. Click on a 'File Content Information' tab, md5 sum is displayed
5. The md5 hash string can now be copied elsewhere

Key words found that are associated with the program/file.

Here is the list of keywords obtained by the tools. The list is obtained by the Linux 'strings' command and the keywords from the file that *overlapped* the CamShell.dll are

[5] According to 1 out of 2 forensics tools: FTK did not see this file, while Autopsy/Sleuthkit did see it

[5] According to 1 out of 2 forensics tools: FTK did not see this file, while Autopsy/Sleuthkit did see it

shown separately. Miscellaneous non-printable characters are also removed where noted. A full list of strings is provided in the Appendix B.

The command used to obtain the data is:

$ strings CamShell.dll > CamShell.dll.str

From the set of obtained list of strings we can conclude:

1. The top of the file really is overwritten by an HTML file and does not look like typical Windows DLL file
2. The remained looks like a Windows program (references to 'shell32.dll'), possibly written in Visual Basic (due to references to 'MSVBVM60.DLL', 'VBRUN', and 'C:\My Documents\VB Programs\Camouflage\')
3. We can compile a set of characteristic keywords that can be used by future investigators to look for traces of this software on disk. Such list will include keywords such as:
   a. CamouflageShell
   b. CamShell
   c. Camouflage

   these strings are certain to characterize the DLL file, distributed as a part of a Camouflage package and are unlikely to be see in other software due to their close relation to camouflage code.

Forensic Details:

What is the name of the program used by Mr. Leszczynski?

Camouflage v. 1 by Twisted Pear Productions, www.camouflagesoftware.com (dead) or http://camouflage.unfiction.com/Download.html (active 11/15/2004)

What type of program is it?

This program is a type of **steganography** software. Some background information on steganography follows**.**

> Steganography is a method to "to embed information for proving authenticity and authorship and for preventing non authorized entities from reviewing the hidden information. This form of information hiding is known as steganography. Derived from Greek words, *steganos* (meaning secret or hidden) and the *graphy* (meaning drawing or writing). The word steganography means the ability to hide information using a form of drawing or writing."

The above information is quoted from: "Steganography-based Forensics Techniques Using Encase 4.0" by Terrence V. Lillard. See additional useful references in the references section.

Here is how the program's authors describe it:

> Camouflage allows you to hide files by scrambling them and then attaching them to the file of your choice. This camouflaged file then looks and behaves like a normal file, and can be stored or emailed without attracting attention.
>
> For example, you could create a picture file that looks and behaves exactly like any other picture file but contains hidden encrypted files, or you could hide a file inside a Word document that would not attract attention if discovered. Such files can later be safely extracted.
>
> For additional security you can password your camouflaged file. This password will be required when extracting the files within.

The above excerpts is from: "Camouflage Software README.txt file v.1.2.1" (obtained from the above site and copied in *C:\Program Files* by the installation routine)

Such programs will likely serve no benign purpose in the company and discovering it serves as a conclusive evidence of some abuse.

While there is a small chance that the program is installed with legitimate purpose, this specific example of its use is clearly illegitimate since confidential information was hidden within the file **and** the note (also hidden) explained that it was intended for sale.

What is it used for?

It is used to hide files within other files, so that the file hidden within can be secretly copied with the host file. The program encrypts the specified data file with a given password and then embeds it into another file (one of the allowed types is required).

We know the above from using the program on the analysis machine and from comparing the recovered DLL file to the one installed with the program.

When was the last time it was used?

From program timestamps, it appears that it was used on 04/26/2004. Unfortunately, FAT file system does not allow for more specific last use determination for files (see above comment about timestamp limitations).

We obtained the MAC times from the forensic tools and can make the above statement

with high degree of certainty.

Include a complete description of how you came to your conclusions, using the forensic analysis methods that were discussed in class.

This process is described above. To summarize, we identified the program by name, searched on the web and located what looked like the same program. We then proceeded to compare the DLL component of the downloaded program to the recovered DLL file. Further analysis, was performed by using the program on the analysis system. It revealed that it can be used to unhide the files hidden by the alleged perpetrator.

You should also include a step-by-step analysis of the actions the program takes and how it works in this section.

Here is what the program does:

1. Takes a document or other file selected by the user on the disk (by right clicking and choosing 'Camouflage')
2. Asks the user to pick another document or file to be hidden within the first one
3. Asks for a password to "prevent" others from uncamouflaging the hidden file
4. The resulting document is embedded within the file and saved on disk

Note that the above information was obtained from using the program on the analysis machine and not from the forensic evidence obtained.

Program Identification:

Locate the program (Look up the source code on the Internet.)

At the time of this writing, the program can be downloaded from http://camouflage.unfiction.com/Download.html.

No source is available. Only binary installation package can be obtained.

Compile and examine the program and compare the results to demonstrate that the program you download is the exact program that was used. Your comparison must to include a comparison of MD5 hashes and how you arrived at them. Include a full description of your research process and the methods used to come to your conclusions.

The version 1.2.1 of the program was downloaded for the investigation. Here is what was performed to positively identify the used program:

Preliminary conditions:

1. The floppy disk in question only contained the *dll* library module for the

Camouflage program, and not the program itself (thus it cannot be executed)
2. The dll file appears to be overwritten by another file (thus it cannot be recovered in full)

Actions taken to analyze the program:

1. Program downloaded
2. Program installed on the analysis machine
3. MD5Sum of *dll* module is computed (md5sum of the dll is 4e986ab0909d2946bed868b5f896906f)
4. The above MD5Sum is compared to the md5sum of the program restored from the floppy (see above tables, md5sum = 6462fb3acca0301e52fc4ffa4ea5eff8
5. **No match** was observed
6. It was explained by the fact that CamShell.dll appears to have been overwritten by another file (HTML document) on the diskette, thus the checksums don't match (see above for string comparison and Appendix B for a full list of strings)
7. However, as we concluded above, many of the strings observed within the dll are the same (see data excerpt above) across the downloaded and recovered dll files. The number of those string matches as well as the same size of the dll files allows us to conclude that it is indeed the same program and likely even the same version.

Legal Implications:

If you are able to prove that this program was executed on the system, include brief discussion of what laws (for your specific country or region) may have been violated, as well as the penalties that could be levied against the subject if he or she were convicted in court.

At this point, we can prove that the program was executed since we have results of such execution (a camouflaged file) were detected. Camouflaging the file requires prior execution of the camouflage program. In addition, there is an attempt to get the information out of the secure facility and (as it seems very likely from the observed behavior) into the hands of competitors. Such act will likely break:

Applicability of laws will center on:

- unauthorized access to information
- intention to defraud

1. Laws
    a. Federal
        i. The Electronic Communications Privacy Act of 1985, 18 U.S.C. Section 2510 (due to unauthorized information access to information)
        http://assembler.law.cornell.edu/uscode/html/uscode18/usc_sec_18_00002510----000-.html
            1. *Example applicable sections:*

a. *"intentionally discloses, or endeavors to disclose, to any other person the contents of any wire, oral, or electronic communication"*

ii. Computer Fraud and Abuse Act of 1986, 18 U.S.C. 1030 *(due to unauthorized information access to information)* http://assembler.law.cornell.edu/uscode/html/uscode18/usc_sec_18_00001030----000-.html or http://www.panix.com/~eck/computer-fraud-act.html

1. *Example applicable sections:*
   a. *"knowingly access a computer without authorization"*
   b. *"obtained information that has been determined… to require protection against unauthorized disclosure"*

iii. Likely trade secret laws, if the misappropriated information is classified as a trade secret

b. State (assuming New Jersey) http://nsi.org/Library/Compsec/computerlaw/Newjerse.txt

i. The New Jersey Code Of Criminal Justice, Statute Title 2c, Subtitle 2. Definition of Specific Offenses, Part 2. Offenses against Property, Chapter 20. Theft and Related Offenses, Ii. Computer-Related Crimes *(due to unauthorized information access to information)*

1. *Example applicable sections:*
   a. *"purposely or knowingly and without authorization …takes or destroys any data, data base, computer program…"*
   b. *"Accesses or attempts to access any computer, computer system or computer network for the purpose of executing a scheme to defraud…"*

2. Policies (e.g. http://www.odu.edu/webroot/orgs/ao/po/polnproc.nsf/pages/index)
   a. Corporate security policies related to
      i. Information Access and Data Classification
      ii. Access to Computer Resources
      iii. Unauthorized software installation and use

At this point, there is no way to estimate the punishment that he can receive, but it will likely include a large fine and possibly a prison sentence. He will certainly not be working for the company.

If you are unable to prove that this program was executed, discuss why proof is not possible. If no laws were broken, then explain how the program use may violate your organization internal policies (for example, an acceptable use policy).

We consider that having results of program execution is sufficient to conclude that the program was indeed executed, provided there is no other way to arrive at those camouflaged files.

Additional Information:

References

1. Suspicious program research
    a. Camouflage software documentation
2. File system references
    a. "Using FAT12 in Windows XP Professional" http://www.microsoft.com/resources/documentation/Windows/XP/all/reskit/en-us/Default.asp?url=/resources/documentation/windows/xp/all/reskit/en-us/prkc_fil_yksz.asp
    b. "How are file and folder date/time stamps affected by a copy and move?" http://www.jsiinc.com/SUBI/tip4100/rh4154.htm
3. Forensics tool documentation
    a. AccessData FTK guide
    b. Autopsy/Sleuthkit guide
4. Steganography resources
    a. "Steganography-based Forensics Techniques Using Encase 4.0" by Terrence V. Lillard
    b. "Information Hiding – The Art of Steganography" GSEC Practical by Asha A. Patel
    c. "Current Steganography Tools and Methods" GSEC Practical by Erin Michaud
    d. "(easily) Breaking a (very weak) steganography software: Camouflage" (http://www.guillermito2.net/stegano/camouflage/)
5. Legal resources
    a. FIRST's "Computer crime laws, listed by state" http://www.alw.nih.gov/Security/FIRST/papers/legal/statelaw.txt
    b. "TITLE 2C. THE NEW JERSEY CODE OF CRIMINAL JUSTICE" http://nsi.org/Library/Compsec/computerlaw/Newjerse.txt
    c. "Computer Fraud and Abuse Act"
    d. "The Electronic Communications Privacy Act"
    e. ODU "University Policies and Procedures" http://www.odu.edu/webroot/orgs/ao/po/polnproc.nsf/pages/index
6. General forensics references
    a. Wietse Venema and Dan Farmer "Forensics Discovery" AWL, 2005
    b. Anton Chuvakin and Cyrus Peikari "Security Warrior" O'Reilly, 2004

Part II Forensic Analysis on a System

Quote from the GCFA assignment: "For this assignment, you must document your actual investigation of a potentially compromised system.…"

Executive Summary

Infected system was investigated after the system administrators tried to remove the malware specimen that was slowing the system and causing excessive network connectivity. Multiple infections and other security problems were discovered during the forensics analysis. The machine was infected by a scanning worm that exploited known vulnerabilities present due to its unpatched status.

Synopsis of Case Facts

System admins[21] monitoring a system at a remote office of *Example.com* have observed the following suspicious behavior on one of the workstations:

- Slow system response for local programs
- Slow networking on the same LAN segment as the system in question
- Extraneous processes running even when no user is using the systems

A system was suspected to be infected, although the antivirus software was installed and was believed to be running. A review of antivirus logs from system has revealed that a suspicious process was detected, but not stopped by the antivirus solution. In addition, a log review of network security monitoring has revealed that the system is a source of significant scanning activity. At that stage the machine was disconnected from the net and its business use was frozen.

The incident was then relayed to a local makeshift "incident response team", consisting of a more qualified system admins as well as other IT professionals (referred to as "local team"). It has discovered that the system was configured for antivirus and system updates, but neither was performed due to low disk space on a system partition (C:\). The "incident team" has tried to manually eliminate the malicious software off the system, working from the antivirus logs.

At some point they decided that they need a more qualified help and referred the incident to a corporate team. However, all the actions will have to be done remotely as the central team was not able to physically reach the system or have it shipped for analysis. Thus, the data collection was performed by the local team under the guidance of the professional central incident team.

The following plan was formulated:

1. Supply the local team with data collection tools for volatile information and image collection
2. Try to collect the volatile info (likely already tampered by the local team "investigation")

[21] We use this common shorthand term 'admin' for system administrator.

3. Collect the disk image (likely tampered by the above as well) from all or at least some (system only) disk partitions for analysis
4. Attempt to preserve the original disk drive if possible[22]
5. Transfer the image to the location where the corporate team can analyze it
6. Transfer the appropriate log records and collected volatile data to a central location in combination with the disk images
7. Query local security infrastructure (such as IDS and firewalls) for log data and use it for discovering the scope of infection[23]
8. Analyze the system images in order to discover the precise scope of the infection/compromise

To address the chain of custody concerns, the following measures were taken:

1. Forensics analysis and evidence storage machines were hardened using industry "best practices"
2. Only those involved investigation were allowed to access the systems
3. All the evidence was stored only at the corporate site (where the forensics team operated)
4. Digital evidence temporary stored at the remote site where the system was located was securely deleted

Unfortunately, the original system was wiped (securely formatted), rebuilt and returned to production[24].

Describe the system(s) you will be analyzing:

This section covers some of the background information about the acquired system.

1. Where did you acquire the system?

The system images were acquired by the local incident response team as described above. The actual hardware stayed at a remote location and was not present in the forensics lab. Understandably, this can cause the investigation results to be challenged in court. The investigators have made an effort to collect all possible details about the hardware.

2. What is/was it used for?

It was used for running business reports from a customer database and intranet sites as well as for VPN access to remote partner sites. Thus, the system had connectivity to the internet as well as to the local LAN.

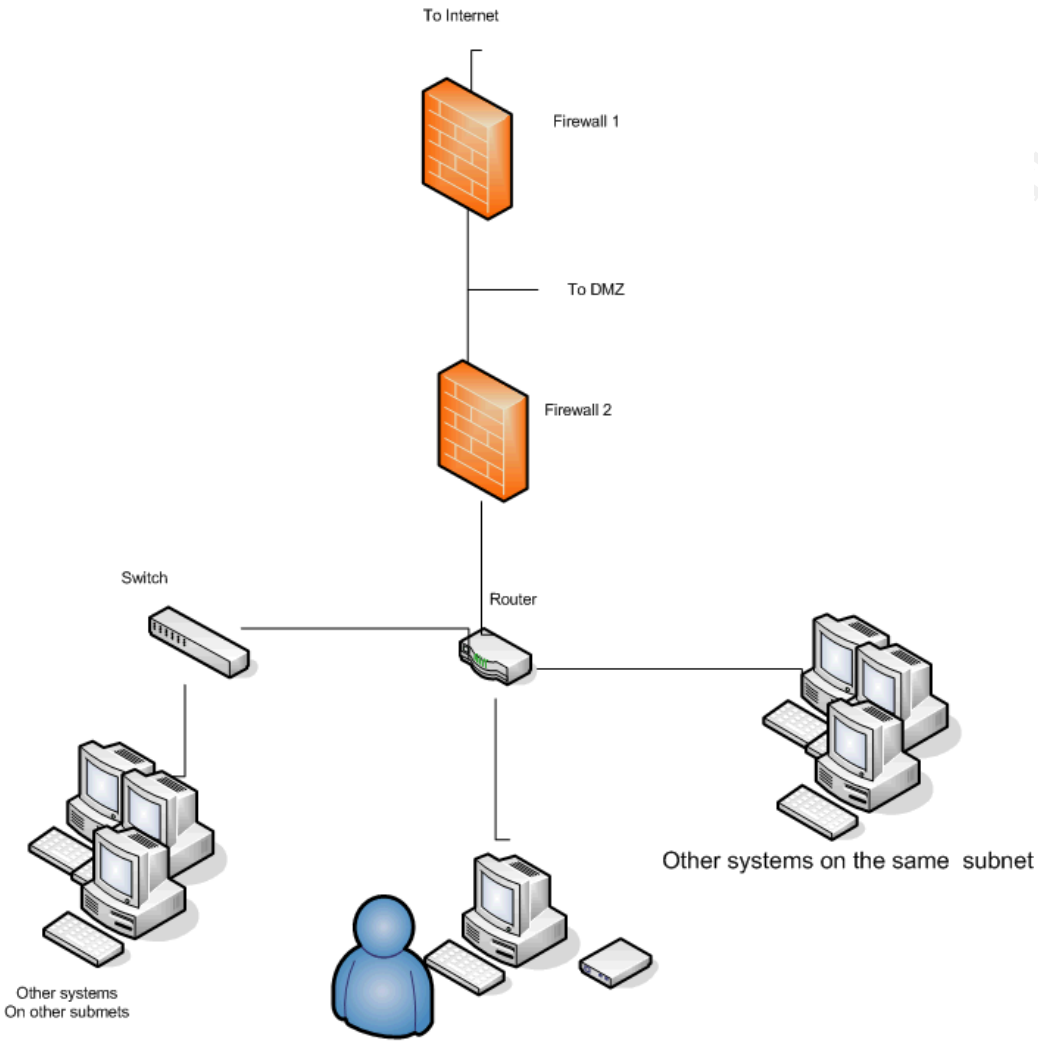
---

[22] As noted later, this turned out to be impossible

[23] This step is not documented here in detail since it is beyond the scope of the practical

[24] We realize that this step goes contrary to usual investigation 'best practices' (and even common sense), but sometimes other matters override the above. That might be the single biggest obstacle to the case, if litigation was proposed

Here is how it was connected:



3. What is the configuration of the system (OS, network)?

Windows 2000 Professional, SP 4, single network card on a shared LAN (shown in detail on the above picture), IP address 10.13.1.2

4. Include any other information you feel may be necessary to perform the investigation

The system volume (C:\) was full. Admins reported various signs of suspicious activity.

Hardware:

No physical items were seized (thus no evidence tags are provided). Admittedly, that will

interfere with possible court case (which was not planned in any case[25]). The administrator team has collected the images based on the instructions from the forensics team, making an effort to preserve the chain of custody (as described above).

Here is the collected information

- Case ID: 0002
- Item ID: 0001
- Description: computer system
- Host ID: 0001
- Seized from: remote office of Example.com

Detailed system description:

- System: Dell Dimension 4300
- CPU: Intel Pentium 4 1.7GHz
- RAM: 512 MB Generic brand
- Hard disk 1: Western Digital WD1000 Caviar 40GB (partitioned into 2 volumes: C:\ sized at 4GB and D:\ sized at 36GB) , serial number: WD088720
- Hard disk 2: Western Digital 10GB (E:\ of 10GB), serial number: WD056024
- Generic DVD-ROM drive, serial number: 797967576
- Toshiba CD-RW drive
- Generic floppy drive
- One generic network interface cards (100MBit Ethernet)

Image Media:

Date/time for data collection: 16:04 on Fri Jul 02 2004

An image was obtained using the forensic version of the 'nc' command run from a floppy.

The team used a version of NC enhanced with:

- built-in md5 checksum for verification
- disk imaging
- compression module (gzip)

The version was obtained from: http://users.erols.com/gmgarner/forensics/ It was supplied on a floppy disk to the administrator team on the remote site where the system was physically located.

[25] Forensics and incident response best practices call for treating every case as an a potential court case, but real-life often interferes with it

The procedure to follow for image collection was defined as:

1. Choose a secure and virus-clean system on the same network segment that has at least 4 GB of free space; it will be used to store the disk images from the potentially compromised machines
2. Insert floppy disk with forensic tools provided by the investigator team
3. Make sure that the personal firewall is either disabled or will allow 'nc' process to become a server (for the reception of images over the network)
4. Click on Windows Start button and then choose 'Run'
5. Type 'cmd.exe' in the resulting dialog box, a command prompt window will appear
6. Run a 'nc' command by typing the command specified below in the command prompt window
7. Reconnect the potentially infected system to the local network, making sure that no Internet connectivity is possible
8. Insert floppy disk with forensic tools provided by the investigator team
9. Click on Start and then choose 'Run'
10. Type 'cmd.exe' in the resulting dialog box, a command prompt window will appear
11. Run a 'nc' command by typing the command below in the command prompt window

Specifically, the commands mandated by the forensics team and executed by the administrator team on the remote site were:

On the collection server system

nc -v -n -l -p 3333 -c zlib -O myimage.img.gz

This command will do the following: start the network reception ('-l' flag) of the incoming file on TCP port 3333 ('-p 3333'), assume that the arriving file will be compressed ('-c zlib') and it should be saved as '*myimage.img.gz'* (due to *-'O myimage.img.gz'*). An increased level of debugging messages is also desirable ('-v' is set). Finally, '-n' options set 'nc' to use IP addresses instead of DNS names.

On the victim system:

nc -v -n -c zlib -csum md5 -l \\-*C: 10.16.10.140* 3333

This command will do the following: start the image collection process (image from C:\ drive as indicated by the '*-l* \\-*C:*'), compress them image using the zlib compression (as indicated by '-c zlib'), compute the MD5 checksum ('*-csum md5*') and open the port to the remote system with IP address *10.16.10.140* and TCP port 3333. The process need to be done with outputting an increased amount of debugging information to make sure that no problems has surfaced (enabled due to '-v' option). Finally, '-n' options set 'nc' to use IP addresses instead of DNS names.

The options utilized aimed at saving the bandwidth (compression) and well as at leveraging the tool's built-in checksum verification. The latter feature assures that the image is transferred without modification and the checksum is available to the investigators.

The images was then checksummed right after reception:

$ md5sum myimage.img.gz

yielding a checksum of compressed image:

634086cdd499e14f73cab7b629659870

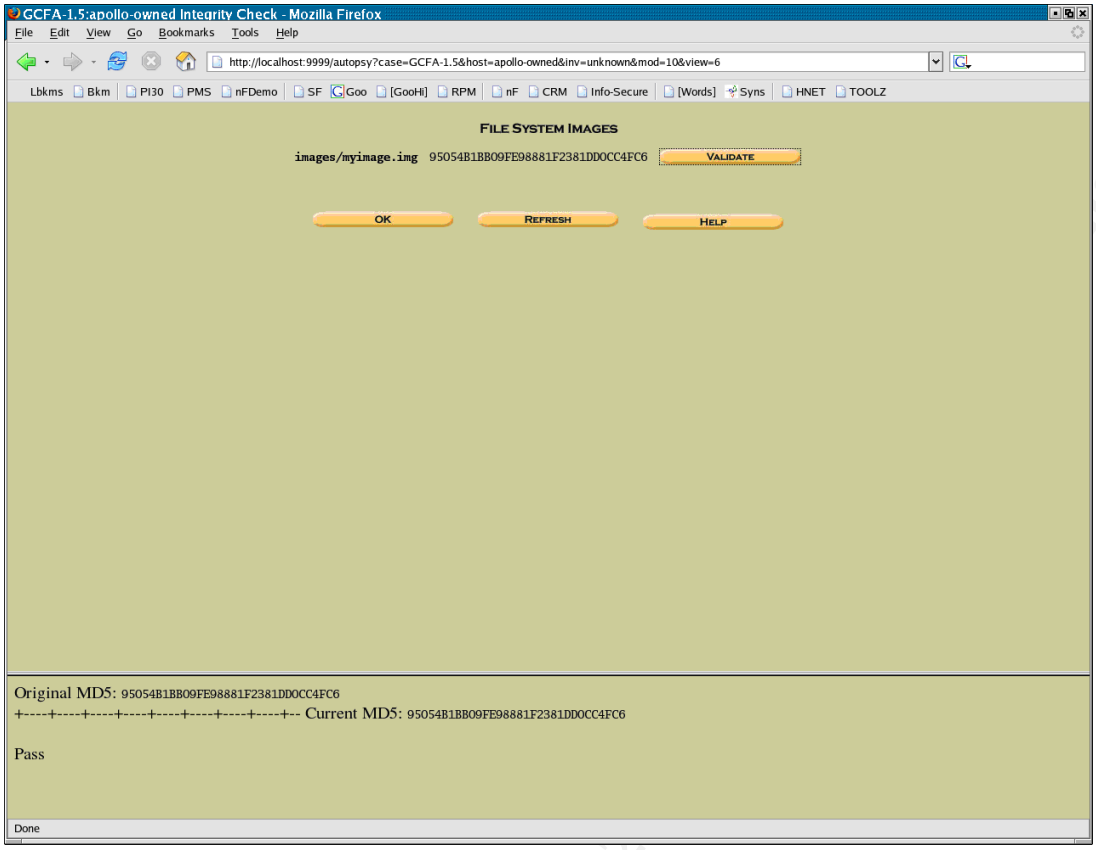
This matched to what was reported by the 'nc' during the data transfer.

The image was then uncompressed and the checksum of the uncompressed one was taken by:

$ gunzip *myimage.img.gz*
$ md5sum *myimage.img*

It was found to be equal to: 95054B1BB09FE98881F2381DD0CC4FC6

The image was then transferred to a central location for the investigation to begin. The uncompressed image of a C:\ drive totaled 4GB (about 1GB compressed). Uncompressed image was also checksummed and then verified by the forensic tools used (Autopsy/Sleuthkit). See this for the evidence of image verification by the forensic tool:



In addition, IRCR toolkit (from http://www.securityfocus.com/tools/2024) was used to collect volatile data. The kit was supplied by the remote administrator team on another floppy disk and the results collected and sent for the investigation.

Here is how the information was collected. This collection procedure was given to the administrator team:

1. Insert the floppy with IRCR into the potentially infected system
2. Click Start, then choose Run
3. Type *a:\IRCR\ircrnt.exe*
4. Wait for data collection to finish (up to 15 minutes and possibly more)
5. Extract the floppy and send all the files on it to the investigator team

IRCR kit collects the following information from a system:

- File Shares
- Global Groups
- Local Groups
- Sessions
- Shared Resources
- Network Services
- Network Users

- Network Connections
- Routing Table
- IP Configuration
- NetBIOS Session
- Protocol Statistics
- Address Resolution
- Hidden Files
- Timestamp File Listing
- Graphical File Listing
- Event Log
- Application Log
- System Log
- Security Log
- Start Up Files
- Detailed Services Report
- Registry Information
- Banner
- Streams

(this list is extracted from the report produced by the IRCR). The collection is performed through a variety of methods, such as exporting logs, querying the registry, and running system utilities.

Here is some of the system information as collected by IRCR (partly sanitized):

```
Caption: Microsoft Windows 2000 Server
Manuf: Microsoft Corporation
BootDevice: \Device\Harddisk0\Partition1
System Dir: C:\WINNT\system32
Organization: Example.com
BuildNum: 2195
Build: Uniprocessor Free
Version: 5.0.2195
CSDVersion: Service Pack 4
Locale: 0409
WinDir: C:\WINNT
TotMem: 391404 bytes
SerNum: 23544-456-3425367-72475
```

Network configuration was also collected:

```
IP CONFIGURATION

Windows 2000 IP Configuration
Host Name . . . . . . . . . . . . : ownedbox
Primary DNS Suffix . . . . . . . :
Node Type . . . . . . . . . . . . : Hybrid
IP Routing Enabled. . . . . . . . : No
WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter {2CAA7452-4B03-41C2-A617-256727462DC3}:

Connection-specific DNS Suffix . :
Description . . . . . . . . . . . : NOC Extranet Access Adapter
Physical Address. . . . . . . . . : 33-44-55-33-44-00
DHCP Enabled. . . . . . . . . . . : No
IP Address. . . . . . . . . . . . : 10.13.1.2
Subnet Mask . . . . . . . . . . . : 0.0.0.0
Default Gateway . . . . . . . . . :
DNS Servers . . . . . . . . . . . :

Ethernet adapter Local Area Connection:

Media State . . . . . . . . . . . : Cable Disconnected
Description . . . . . . . . . . . : 3Com 3C920 Integrated Fast Ethernet Controller
(3C905C-TX Compatible)
Physical Address. . . . . . . . . : 45-C0-3F-69-8D-6E
```

Uptime

up 2 days 06:17 (since Wed Jun 30 10:11:35 2004)

This indicated that apparently the system was rebooted by the administrator team during their initial investigation.

Other collected information will be reviewed below (and provided in full in Appendix C)

Media Analysis of System:

First, we describe what we used to perform the investigation.
The analysis system used was:

- Linux RedHat 9
- Intel CPU 1.8 GHz
- 1 GB RAM
- 60 GB Hard disk

The secondary analysis system used as:

- Windows XP SP2
- Intel CPU 1.4 GHz
- 512 MB RAM
- 40 GB Hard disk

Forensics tools used were:

1. Autopsy/Sleuthkit (see reference in **Part I**)
2. Standard Unix/Linux file system tools (see references in **Part I**)
3. AccessData FTK - (see reference in **Part I**)
4. AccessData RegistryViewer (http://www.accessdata.com) – demo version with feature limitations
5. IRCR (Incident Response Collection Report) (http://ircr.tripod.com/)

Here is the table summarizing the information about tools:

Vendor	**Name**	**Purpose**	**Why used**
Brian Carrier	Autopsy/Sleuthkit	Forensic toolkit	Analyze the disk image
RedHat Linux	Misc utilities	System tools	Perform various data analysis tasks on files and images
AccessData	RegistryViewer	Review Windows registry files collected from the system	Analyzed malicious registry changes
AccessData	FTK (Forensic ToolKit)	General purpose forensic toolkit for complete image analysis	Analyze the IE history files
John McLeod	IRCR (Incident Response Collection Report)	Volatile and log data collection from a Windows system	Collect and present data other than disk images

Steps taken to avoid modifying the images:

1. The original image was copied to a backup tape immediately after the transfer from the remote site
2. The analysis copy was periodically compared to the backup
3. The analysis copy was set to be immutable (*chattr +i image)* and write permissions were removed (*chmod a-w image*) so that the image cannot be accidentally modified by users *or* forensics tools
4. The forensic tools used on the image (Autopsy/Sleuthkit) is a standard forensics tool used by many investigators. It has been validated to not modify the images by other investigators. In addition, the image file system permissions prevented its

modification

The system was analyzed for signs of compromise and presence of malware, based on the information in the antivirus logs and other things reported by the administrator team. Note that the analysis performed does not constitute a full analysis of all system irregularities or a comprehensive study of all files on the disk. It was focused around malware and other system irregularities that affect the secure and reliable operation of the system.

Antivirus history

First we looked at the antivirus logs, as the major reported problems were related to malware. Symantec antivirus history file reveals a fun series of virus incidents. The administrator team exported the log from the Symantec anti-virus console (before the rest of the investigation was performed) and send it to the investigating team.

The steps to collect the log were:

1. Click on the Symantec antivirus icon on the toolbar of the system protected by the anti-virus (in our case, this is our potentially infected system)
2. Click on 'Histories', then choose 'Event Log'
3. Click on a button with floppy disk to export the log
4. Put a desired filename in the window
5. Press 'Save'. The log will be exported in the CSV format.

Understandably, this has caused changes to the access time stamps of multiple files, possibly contaminating the evidence.

Investigators then opened it in Excel (screenshots shown below). Also, it could have been extracted from the image directly by searching for common keywords characteristic of Symantec antivirus logging, such as 'Quarantine', ''Action Taken', 'Virus Name' and others seen below).

	A	B	C	E	G	H	J	M
1	Date	Filenam	Virus Name	Action Taken	User	Original Location	Current Location	Scan Type
2	7/2/2004 1:07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
4	7/1/2004 1:07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
5	6/30/2004 19:45	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
6	6/30/2004 19:45	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
7	6/30/2004 19:44	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Realtime scan
8	6/30/2004 1:07	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Scheduled scan
9	6/29/2004 18:29	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual scan
16	5/28/2004 21:55	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual scan

The above excerpt indicates that the system was affected by this backdoor since at least May (as shown by this filtered log sample from Symantec Antivirus). It also shows that the backdoor was not cleaned by the solution. While the log might have been rotated, we can look at a full log from that time period:

	A	B	C	E	G	H	J	
1	Date	Filename	Virus Name	Action Taken	User	Original Location	Current Location	Scan T
14	6/4/2004 20:23	wumgrd.exe	W32.Randex.gen	Left alone in Quarantii	SYSTEM	C:\WINNT\system32\	Quarantine	Defwat
15	6/4/2004 20:23	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\WINNT\system32\drivers\	Quarantine	Defwat
16	5/28/2004 21:55	msrll.exe	Backdoor.IRC.Bot	Left alone	Administrator	C:\WINNT\system32\mfm\	C:\WINNT\system32\mfm\	Manual
17	5/28/2004 21:51	wumgrd.exe	W32.Randex.gen	Quarantined	Administrator	C:\WINNT\system32\	Quarantine	Manual
18	5/28/2004 19:21	svchost.exe	W32.Welchia.B.Worm	Quarantined	Administrator	C:\WINNT\system32\drivers\	Quarantine	Manual
19	5/17/2004 12:06	25227_up.exe	W32.Sasser.C.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
20	5/17/2004 12:06	25227_up.exe	W32.Sasser.C.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
21	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
22	5/17/2004 12:06	TFTP988	W32.Randex.gen	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
23	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
24	5/17/2004 12:06	TFTP1256	W32.Randex.gen	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
25	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
26	5/17/2004 12:06	TFTP1128	W32.Welchia.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
27	5/17/2004 12:06	TFTP328	W32.Welchia.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\	Quarantine	Defwat
28	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
29	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
30	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
31	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\Documents and Settings\D	Quarantine	Defwat
32	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\Documents and Settings\D	Quarantine	Defwat
33	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\Documents and Settings\D	Quarantine	Defwat
34	5/17/2004 12:06	svchost.exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\winnt\system32\drivers\	Quarantine	Defwat
35	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\Documents and Settings\D	Quarantine	Defwat
36	5/17/2004 12:06	WksPatch[1].exe	W32.Welchia.B.Worm	Left alone in Quarantii	SYSTEM	C:\Documents and Settings\D	Quarantine	Defwat

It shows that around that time the system was infected by a Welchia worm that could have deposited the backdoor Trojan. It also indicates that the system presents a 'rats nest' of various malware, only some of which was successfully cleaned (and often after the fact of successful infection and system modification).

In fact, the reference for *W32.Randex.gen* (http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.gen.html#technicaldetails) says that the worm is known to perform "opening backdoor ports and opening connections to predetermined IRC servers and waiting for commands from an attacker. Such IRC activity (IRC connection attempts blocked by the firewall) were in fact detected by network monitoring[26] at some time after the initial May 28 infection:

```
28 Jun 04 08:03:23 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  22   26   195   8738   CON
28 Jun 04 08:04:24 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  5    5    0     475    CON
28 Jun 04 08:05:29 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  6    7    31    399    CON
28 Jun 04 08:06:34 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  5    5    0     432    CON
28 Jun 04 08:08:10 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  13   14   31    1103   CON
28 Jun 04 08:09:10 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  7    7    0     622    CON
28 Jun 04 08:10:11 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  13   14   31    1061   CON
28 Jun 04 08:11:19 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667  6    6    0     595    CON
28 Jun 04 08:12:19 q   tcp  172.xx.yy.103.1066  ->  207.36.231.153.6667
```

This Argus output format shows the connection from out system to remote IRC server (TCP port 6667) and also the amount of data transferred (packets and bytes) as well as the fact that the connection was established.

Thus, we conclude that the system was infected and even attempted to connect to remote sites, since no legitimate IRC client was installed on it.

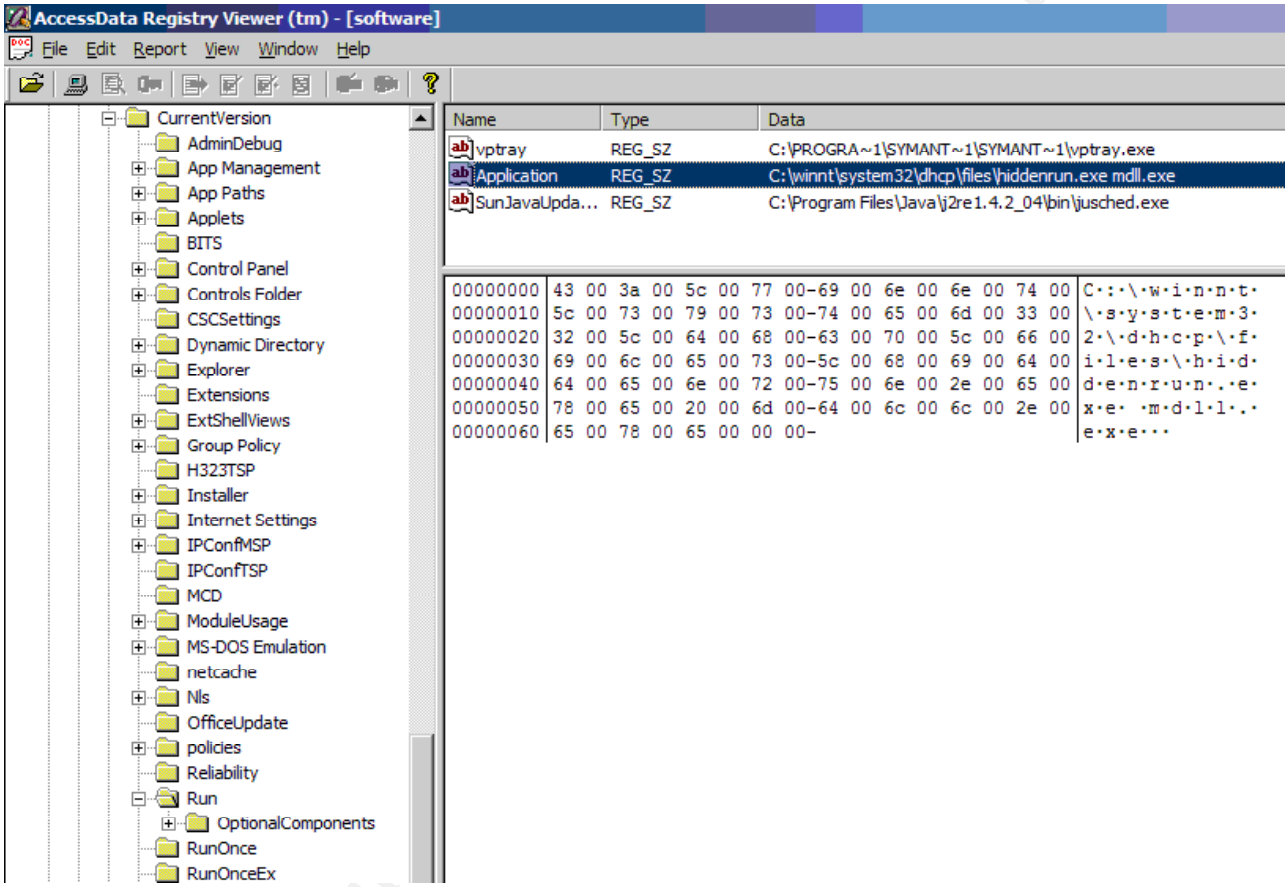
At that point, the command logs would have come handy. Unfortunately, Windows systems do not have any built-in command line shell history collection mechanism.

[26] Via Argus flow monitoring tool

Next, it is logical to look at the registry, that can provide important additional hints about malware and other security problems.

Registry

On a Windows system, registry is usually a treasure trove of useful information. Let's examine it using AccessData RegistryViewer:



We have extracted the folder containing the registry files from the captured image. Here is how we did it:

1. The folder name where the registry is stored *C:\WINNT\SYSTE32\CONFIG*. It contains several files in the "registry format".
2. We copied the files from the above directory on the captured image to our Windows analysis systems
3. AccessData RegistryViewer was used to review them (see above the examination of *C:\WINNT\SYSTE32\CONFIG\Software* in the viewer). To open a registry file, one needs to go to a File menu and choose Open file.

The screen shot shows the important key in Windows registry (CurrentVersion/Run) that is used to make processes start automatically. A malware "mdll.exe" is started using a

"hiddenrun.exe" wrapper using that key (see highlighted line). This is key is always checked when the malware infection is suspected.

Next we will look at the processes and services that start automatically.
The list is shown in Appendix C, section 3. The system contains a huge number of insecure services running, other data collected by the IRCR (also shown in Appendix C) indicates that active backdoors were running at the time of data collection (listening port 2200). Unfortunately, the incident team did not have a chance to run more advanced tools on a live system to associate the ports with running backdoor processes for reliable confirmation. That would have been an important step in the investigation.

New Files

Next we examine file system for modifications to operating system software or configuration.

To do this we look at the Windows system directory (C:\WINNT\SYSTEM32). During the investigation a comparison between a known clean system and the victim system drive is performed to pinpoint new files.

New files were detected in the system directory. Those include:

a. Malware 1 – Sophos/Troj/Tometa-A
   (http://www.sophos.com/virusinfo/analyses/trojtometaa.html)
   - **msrll.exe**
   - **jtram.conf**

b. Malware 2 - McAfee/IRC/Flood.cm (http://vil.mcafeesecurity.com/vil/content/Print100427.htm)
   - **mdll.exe**[27]
   - **hiddenrun.exe**

c. Malware 3 – Symantec/W32.Randex
   (http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.gen.html)
   - **TFTP1380** and others

   These files do not exist on a clean Windows machine, but are present on the victim image.

   We have not performed the full analysis of Windows system file hashes as it was obvious that the system is heavily infected by malware, some of which was not even being detected by the installed antivirus software (some was detected but not cleaned)

   The files were discovered via:

[27] McAfee reports that mdll.exe is a "legitimate IRC client program, detected as IRC/Client" (see http://vil.mcafeesecurity.com/vil/content/Print100427.htm). The antivirus on the system we are investigating has not alerted on the presence of this file

Malware	Discovery Method
Sophos/Troj/Tometa-A	1. Antivirus logs 2. Forensics tools based on file names from logs
McAfee/IRC/Flood.cm	1. Common suspicious registry keys 2. Forensics tools based on file names from registry scan
TFTP	1. Antivirus logs 2. Forensics tools based on file names from logs

The system did have malware running ("msrll.exe") that is documented to include a port listening backdoor, as well as possibly other backdoors. The *netstat* output collected by the IRCR (and shown in the Appendix B) shows that port 2200 (highlighted) was listening at the time of data gathering.

Here is the brief information on the discovered malware

> **Troj/Tometa-A** is a backdoor Trojan for the Windows platform that copies itself to the file *Windows\System\mfm\msrll.exe* and sets a registry entry under:
>
> HKLM\Software\Microsoft\Windows\CurrentVersion\Run\Rll
>
> enhanced drive to point to it so that the Trojan will always be executed on system startup. Troj/Tometa-A will also drop the data file jtram.comf to the folder specified above and listens on port 2200 for incoming commands from a remote user.

The above is quoted from Sophos web site (http://www.sophos.com/virusinfo/analyses/trojtometaa.html)

The other malware discovered ("mdll.exe") can also be used as a backdoor due to its reported IRC communication capabilities (see link above).

Among other things to Windows system was using a FAT file system, that has no concept of *setuid* and *setgid* files.

Another important piece to look at is various history files.

Internet History

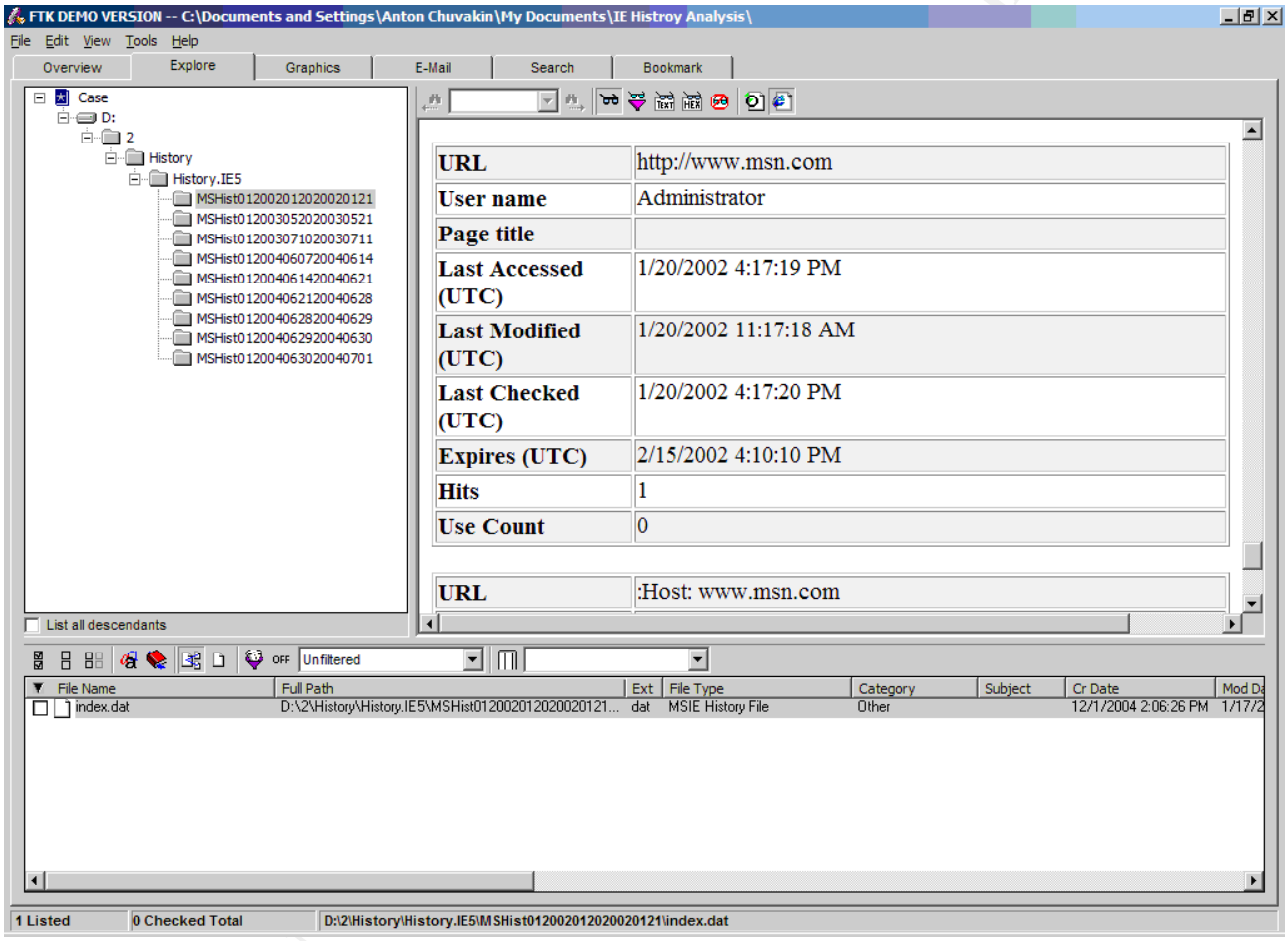
We used AccessData FTK to review the Internet Explorer history files to look for possible infection via IE. Here is the process that was followed:

1. Extract history files and erased history files from the image by using Autopsy/Sleuthkit. To do this one needs to navigate to the right place

where history is stored (*C:\Documents and Settings\Administrator\Local Settings\History*)

2. Copy the data to the Windows analysis machine from the image
3. Add the contents of a folder as evidence to a case in FTK ('Open case', then 'Add evidence')
4. Open the data (see below)

Here is the example data:



The above image shows an Internet Explorer history file record. It indicates that an Administrator user has gone to www.msn.com (in reality, it means that he or she simply opened the IE browser, since www.msn.com is a default home site and the browser goes to it when launched).

The data analysis showed that the users of the system only connected to the intranet sites and to the IE[28] default page (www.msn.com) at the time at and around of the original infection. Thus, the system was apparently not infected through the client (which was a likely possibility), but rather by worm scanning when connected to insecure networks.

[28] Internet Explorer

Event Logs

As everybody knows, Windows NT family system contains 3 main event logs, usually stored in *C:\WINDOWS\system32\config*.Evt:*

- System
- Application
- Security

On our victim system it was actually *C:\WINNT\system32\config* since Windows was installed in a somewhat non-standard location.

While we could have extracted those binary logs from the image (from the above location) and then processed them through the Event Viewer or FTK, IRCR toolkit collected them from the live system and converted them to plain text. Thus, to review the logs we just had to review the text files using any text editor or viewer. So the log files were reviewed manually. Here are some conclusions with excerpts from the actual captured logs:

A. Anti-virus failures: application log contained records similar to this:

RecordNumber: 107
Source: Norton AntiVirus
Computer: OWNEDBOX
Category: 0
Event ID: 16
EventType: 4
Time Generated: Fri May 28 20:03:59 2004
Time Written: Fri May 28 20:03:59 2004
User:
Message:
Download of virus definition file from LiveUpdate server failed. 00000001

Those indicate virus update failures due to log disk space that were the likely reason for system inoperability.

B. Successful clean messages are also logged there. Here is one example:

RecordNumber: 153
Source: Norton AntiVirus
Computer: OWNEDBOX
Category: 0
Event ID: 5
EventType: 1
Time Generated: Fri May 28 23:54:14 2004
Time Written: Fri May 28 23:54:14 2004

User:
Message:

Virus Found!Virus name: W32.Welchia.B.Worm in File: C:\WINNT\system32\drivers\svchost.exe by: Manual scan. Action: Quarantine succeeded :

Virus Found!Virus name: W32.Randex.gen in File: C:\WINNT\system32\wumgrd.exe by: Manual scan. Action: Quarantine succeeded :

Virus Found!Virus name: Backdoor.IRC.Bot in File: C:\WINNT\system32\mfm\msrll.exe by: Manual scan. Action: Clean failed : Quarantine failed :

Those indicate failures and successes of cleaning actions by the antivirus solution.

C. Other

Logs contained a lot of other records related to various antivirus failures (to access, to open files, to update, etc). Some of the more interesting logs were rotated away due to volume. That is a known issue that may hinder an investigation that occurs a long time after the infection/compromise.

Timeline Analysis:

We will use Autopsy/Sleuthkit advanced timeline capability. To do this we did the following steps:

1. Launch Autopsy/Sleuthkit
2. Choose the right case to open
3. Chose the right host within the case
4. Click 'File Activity Timelines'
5. Click on 'Create Data File', follow prompts
6. Click on 'Create Timeline', follow instructions
7. Click on 'View Timeline' ; time line is displayed in browser (this is what is featured in most of the screenshots in this section)
8. One can also export the timeline to a plain text file (this is what is attached to this document)

Timeline Summary:

This system contains files dating from 1995 (likely from Windows distribution) to the date of the permanent system shutdown for investigation (after imaging on July 2, 2004)

Resulting timeline is big (7MB text file) and only excerpts are provided. Full timeline is attached to submission.

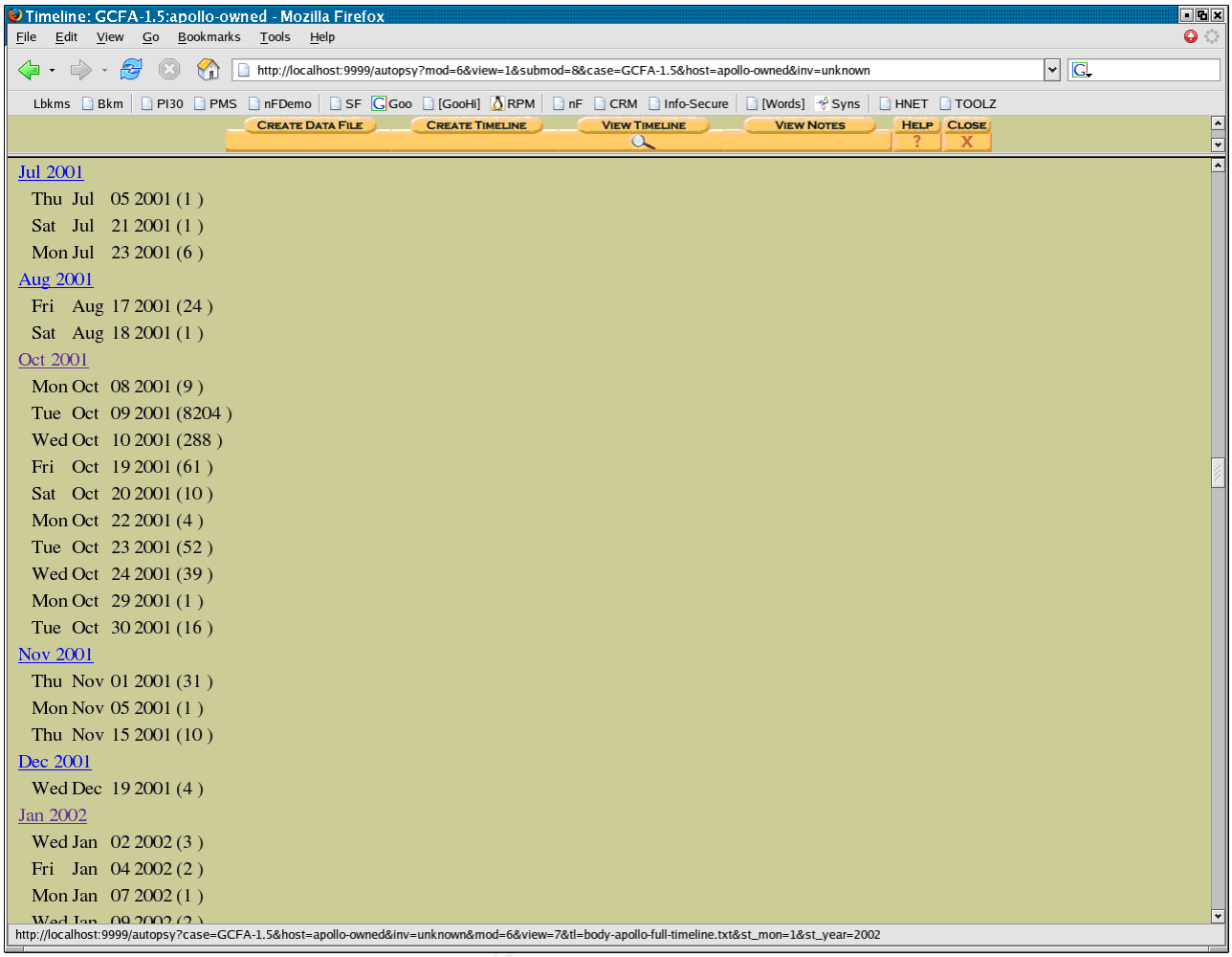
Major events:

- Operating System installed

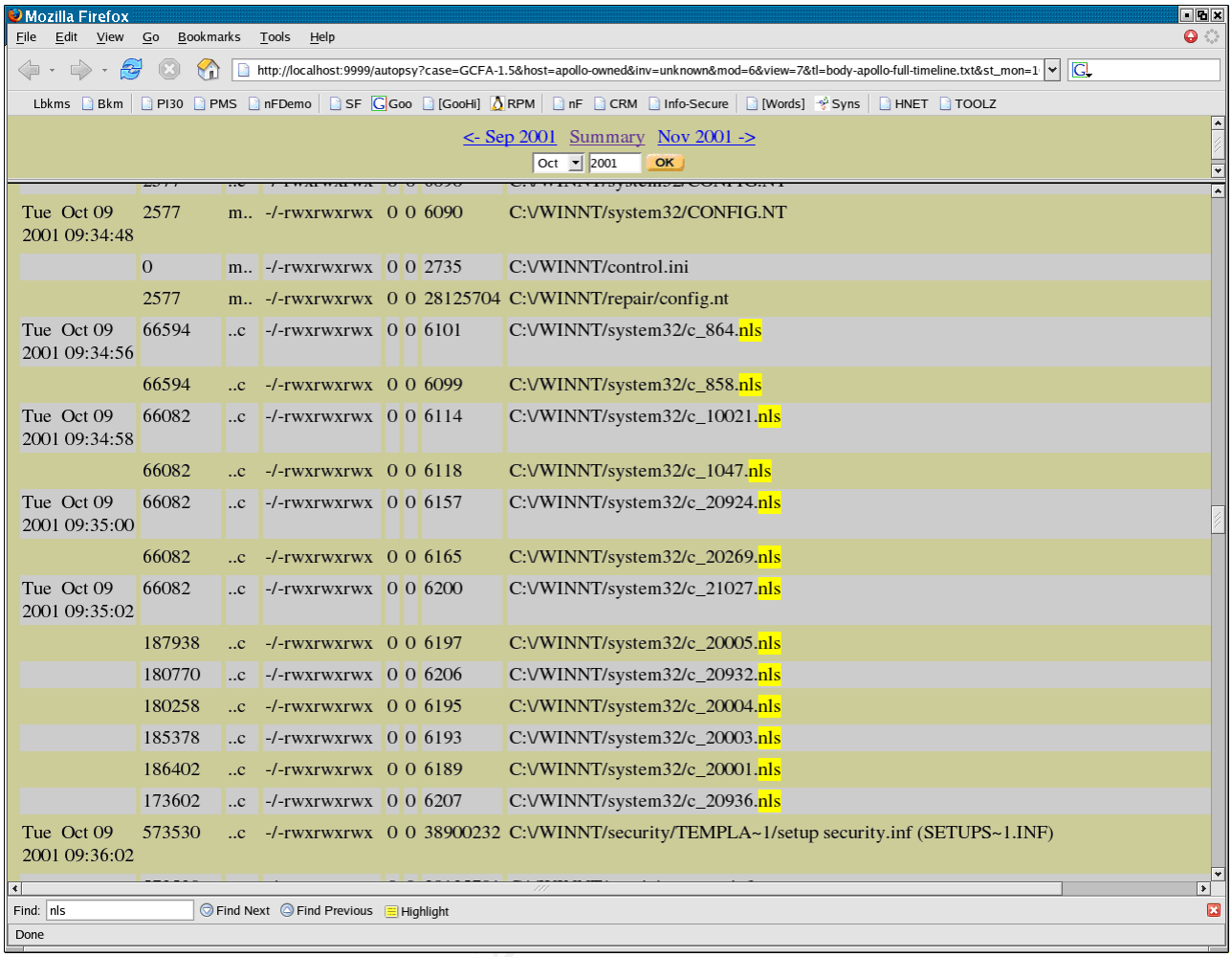
We hypothesize that the Windows was installed in the morning of October 9, 2001. There are several traces pointing to this:

- a large number of file is affected at that date; a lot of files will have modification timestamps pointing to the day of installation since they will created or modified at this date. Examples include various configuration files (*.ini), registry files and many other types of system files as well as temporary files. The same logic applies to application installed on the same day. They will have many files with dates pointing to the time of coding or compilation, but most configuration files will likely be dated with the installation date
- the types and names of those files are the same that changed at the date of install on other reviewed Windows 2K and Window XP systems. We looked at a timeline of a freshly built Windows 2000 system, and a lot of files were modified at the date of installation (the first largest set of such changes if we look from the earliest present dates)

Here is the anomalously large changed file count shown in the Autopsy interface:



More specifically, one example of those files are the *.nls files. On our freshly installed system they were dated with the system installation date (that is the main reason why we chose them as an example). Thus, it is safe to assume that the earliest date of their modification across the whole timeline will be the moment of installation. While the might be updated later, the earliest change of them is the installation date. See the next picture for some of those files (of type .nls)



- System updates

Now let's look at Windows 2000 service pack (SP) schedule and then try to match it to timeline changes in order to verify when the system was updated. We will likely not be able to determine the dates of all the patches, but can hope to pinpoint the dates when service packs were deployed, incurring major changes to the system.

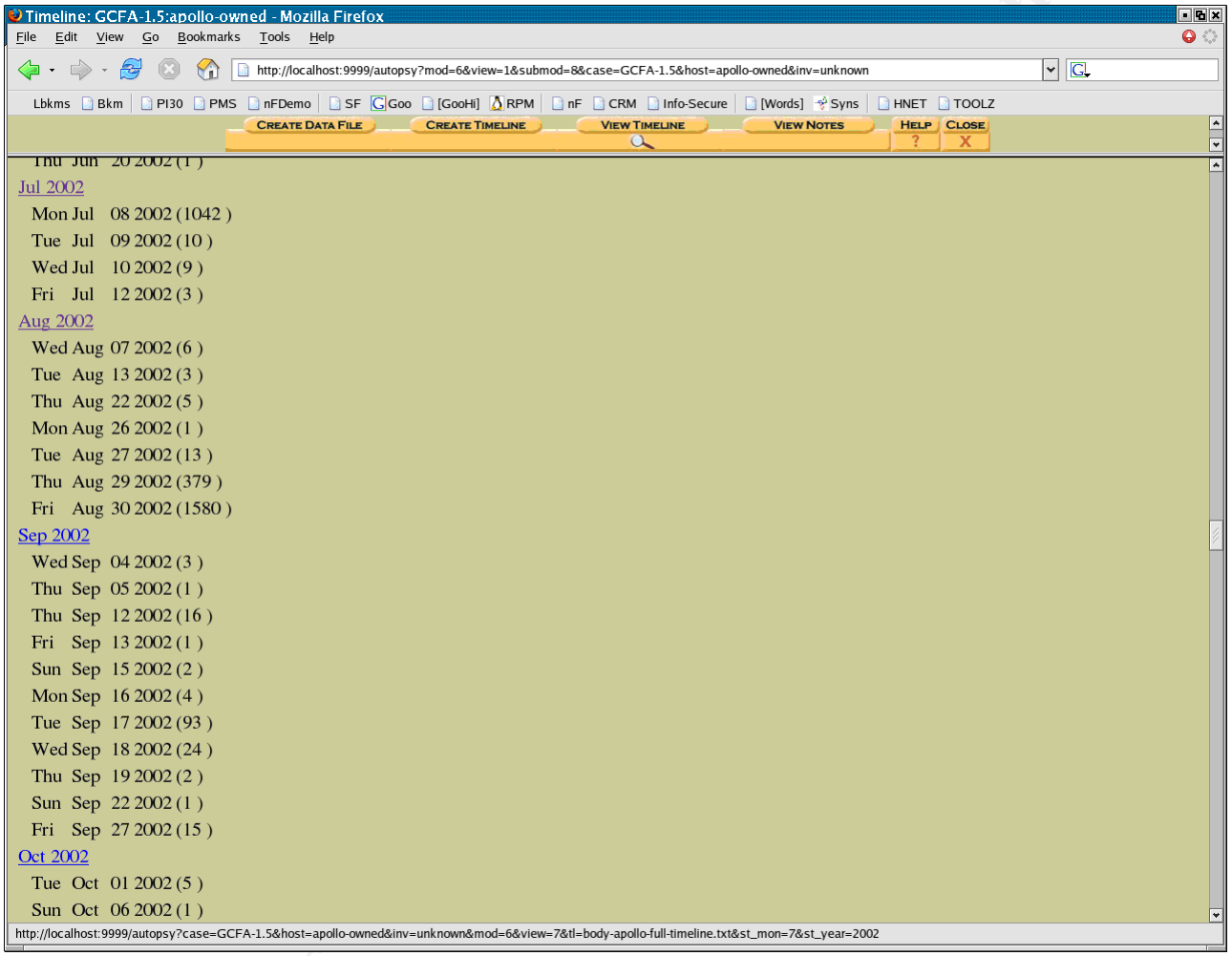
The following SP schedule was collected by searching in Google for "Windows service pack release" and then specifically looking for service packs 1,2,3 and 4.

Pack	Release Date
SP1	Jul 2000
SP2	May 2001
SP3	Aug 2002
SP4	Jun 2003

We hypothesize that the system built in Oct 2001 will have at least SP1 installed and likely SP2 as well. System CDs often come with a service pack already in place and thus

without the need to deploy it separately.

Let's look through the summary timeline for the next period of major file changes. Perhaps not surprisingly, it is August 2002. It looks like the admins did keep up with patches after all.



The above figure shows that compared to adjacent months, a month of August features a large number of file changes, likely indicative of a patch update. Specifically, August 30, 2002 seems to be the day the system was updated.

However, one might argue that some other major software installation might have caused it. But a quick look at the detailed timeline (see attached) indicates that many of the changes concentrate in the C:\WINNT directory. For example, a lot of files in *C:/WINNT/system32* are modified.

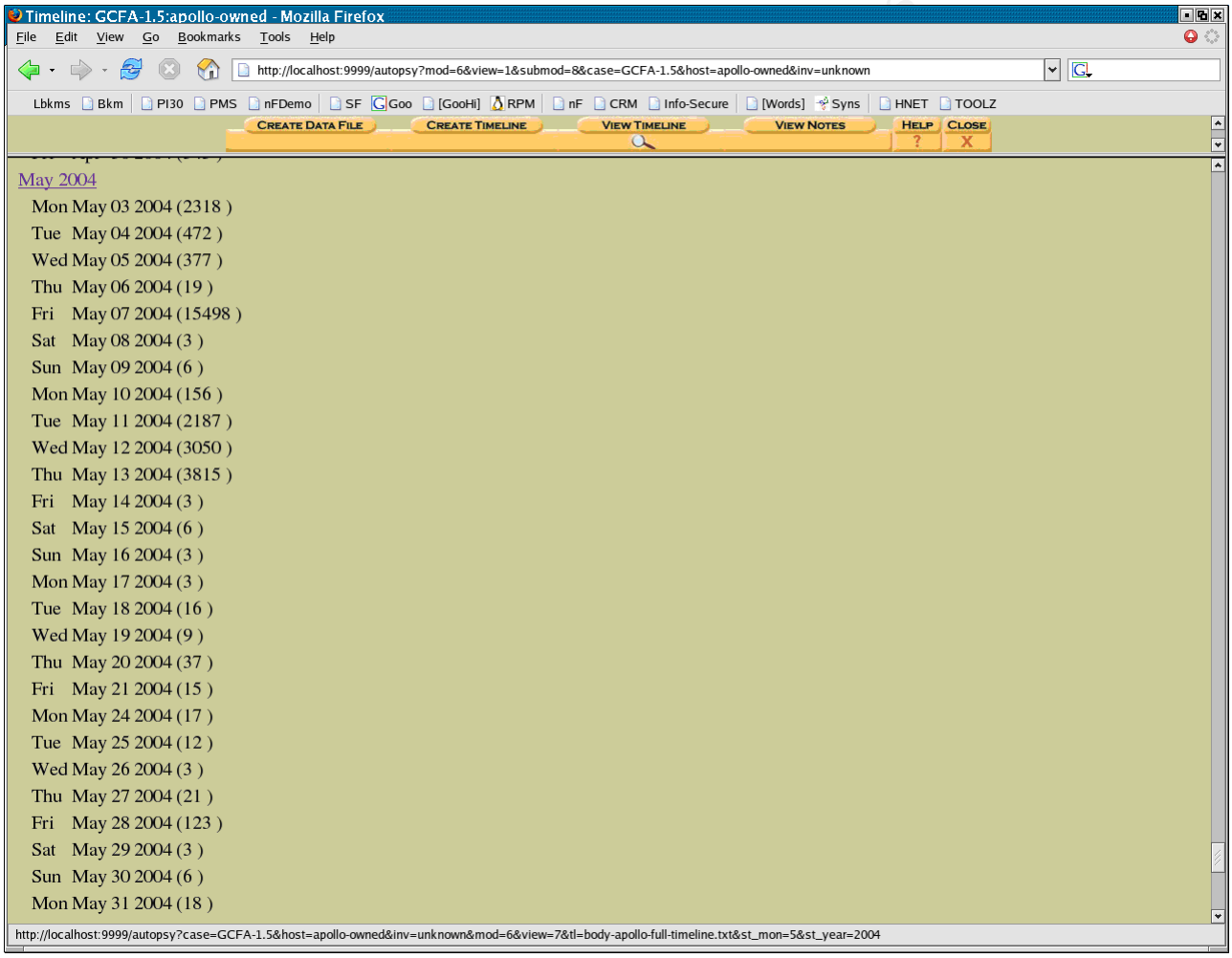
Exactly the same phenomenon is observed in June 2003 (for SP4).

As a result, the final system is patched up to SP4 level (the last Windows 2000 Service Pack), exactly as indicated by the IRCR output above.

Thus, just by looking at the timeline one can make an educated guess about the history of the system.

- Malware infected date

May 2004 was a bad month for this computer system ☺ It looks like it saw some major malware activity (investigated in this report).



The above screen shot shows May 10-13 (as well as May 7) as major file system change dates, similar to the above changes related to service pack installation).

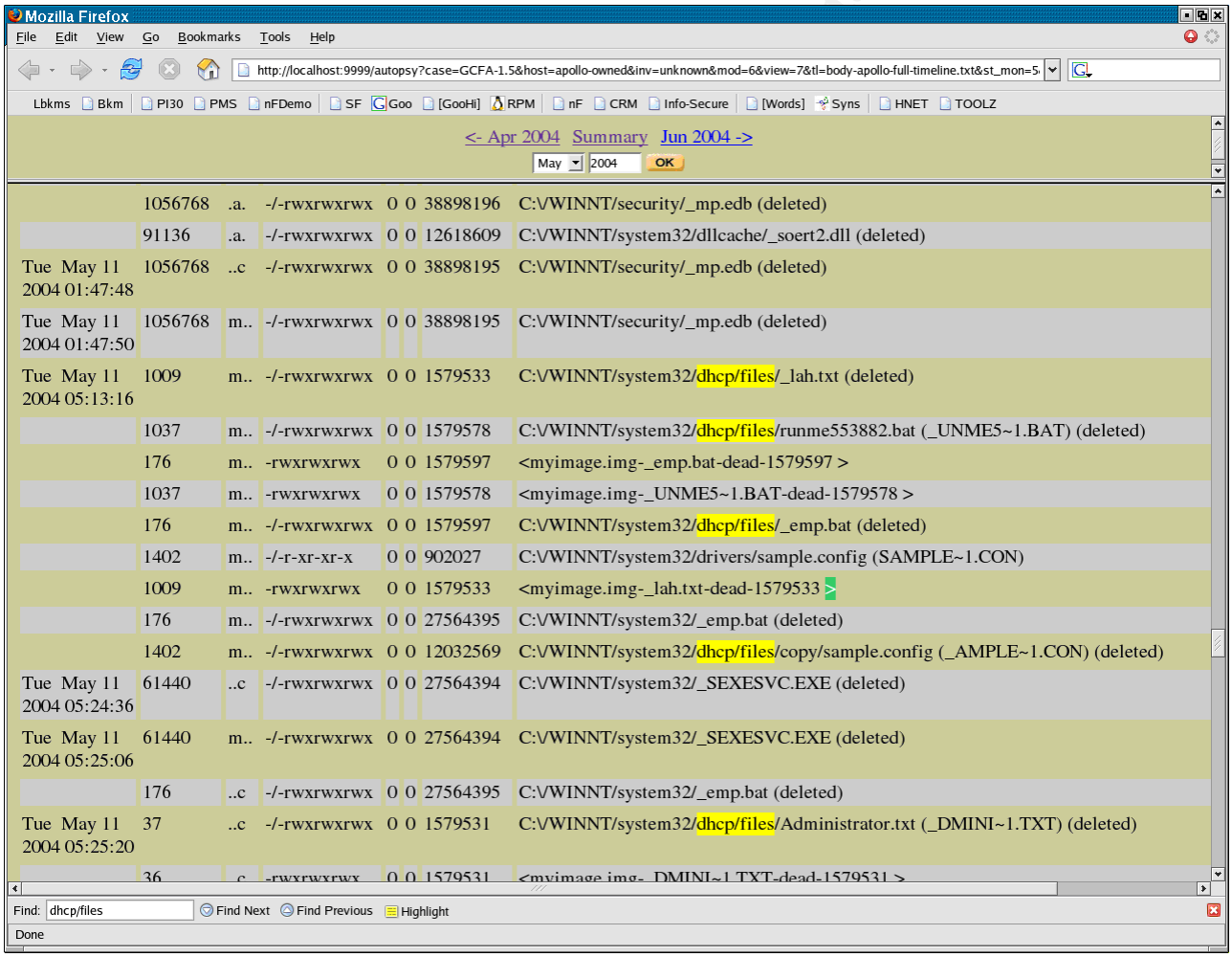
Let's drill down further by looking at this month in detail. While there are lots of changes recorded, to narrow down the presented data we will search for the malware name that we observed in the antivirus logs – 'msrll.exe'. May 10 2004 is when this file name first appears in the timeline log.

Specifically, we see:

```
Mon May 10 2004 16:29:54    41984 m.. -/-rwxrwxrwx 0       0       27564392 C:\WINNT/system32/_080.exe (deleted)
Mon May 10 2004 16:29:54    41984 m.c -/-rwxrwxrwx 0       0       12024325 C:\WINNT/system32/mfm/msrll.exe
Mon May 10 2004 16:29:54    65536 ..c d/drwxrwxrwx 0       0       27564393 C:\WINNT/system32/mfm
Mon May 10 2004 16:29:56    65536 m.. d/drwxrwxrwx 0       0       27564393 C:\WINNT/system32/mfm
Mon May 10 2004 16:30:14    1208 ..c -/-rwxrwxrwx 0        0       12024327 C:\WINNT/system32/mfm/jtram.conf (JTRAM~1.CON)
```

(the above timeline entry is modified by adding the date in front of all the entries for clarity)

Thus we can conclude that the malware was introduced on that date (which correlates with the antivirus logs). Further analysis reveals the following:



The above shows some major activity in the malware-related *C:\WINNT\system32\dhcp\files* (as selected on the image) on the next day (May 11). Specifically, we see files being added, modified and removed. This above image is produced by clicking on the hyperlink related to the month of May on the above view in Autopsy.

Thus, we conclude that timeline analysis can be very handy in tracking the infection history and having other information sources (such as antivirus logs) helps to guide the process much faster.

- Early “investigation” traces

Late June of 2004 shows a lot of file access activity as well as some deletions. It is indicative of an early recovery and analysis activity by system administrators. At the same time, antivirus files have changed, since the admins finally got the updates working.

Specifically, we see several interesting activities related to the incident in the full timeline file:

1. a massive number of access timestamps on various executable files likely indicate an antivirus software full disk scan (schedule and not interactive, since it started at 00:00:00). Such activity is seen on Jun 30

```
Wed Jun 30 2004 00:00:00      67 .a. -/-r-xr-xr-x 0       0       39899653 C:\WINNT/TEMPOR~1/CONTENT.IE5/856ZCH6J/desktop.ini
 Wed Jun 30 2004 00:00:00     6416 .a. -/-rwxrwxrwx 0      0       2395386  C:\WINNT/system32/dllcache/kbdbu.dll
Wed Jun 30 2004 00:00:00      34240 .a. -/-rwxrwxrwx 0     0       2394670  C:\WINNT/system32/dllcache/nserror.dll
Wed Jun 30 2004 00:00:00      43280 .a. -/-rwxrwxrwx 0     0       2395046  C:\WINNT/system32/dllcache/docprop.dll
Wed Jun 30 2004 00:00:00      153872 .a. -/-rwxrwxrwx 0    0       2396567  C:\WINNT/system32/dllcache/vcmshl.dll
Wed Jun 30 2004 00:00:00      18192 .a. -/-rwxrwxrwx 0     0       2395940  C:\WINNT/system32/dllcache/pathping.exe
Wed Jun 30 2004 00:00:00      22288 .a. -/-rwxrwxrwx 0     0       2395553  C:\WINNT/system32/dllcache/mciseq.dll
Wed Jun 30 2004 00:00:00      97040 .a. -/-rwxrwxrwx 0     0       12618880 C:\WINNT/system32/dllcache/clbcatex.dll
Wed Jun 30 2004 00:00:00      76048 .a. -/-rwxrwxrwx 0     0       2395697  C:\WINNT/system32/dllcache/avwav.dll
Wed Jun 30 2004 00:00:00      56592 .a. -/-rwxrwxrwx 0     0       2395864
```

2. some file deletions on Jun 30 seem to indicate the administrator team trying to fight the malware by removing its files

```
Wed Jun 30 2004 19:42:58      47 m.. -rwxrwxrwx 0       0       1579754  <myimage.img-_emote.ini-dead-1579754>
Wed Jun 30 2004  19:42:58     47 m.. -/-rwxrwxrwx 0      0       1579754  C:\WINNT/system32/dhcp/files/_emote.ini (deleted)
```

etc

- System last used (by the administrator team taking an image)

Let’s look at the last entries at the timeline file. They are:

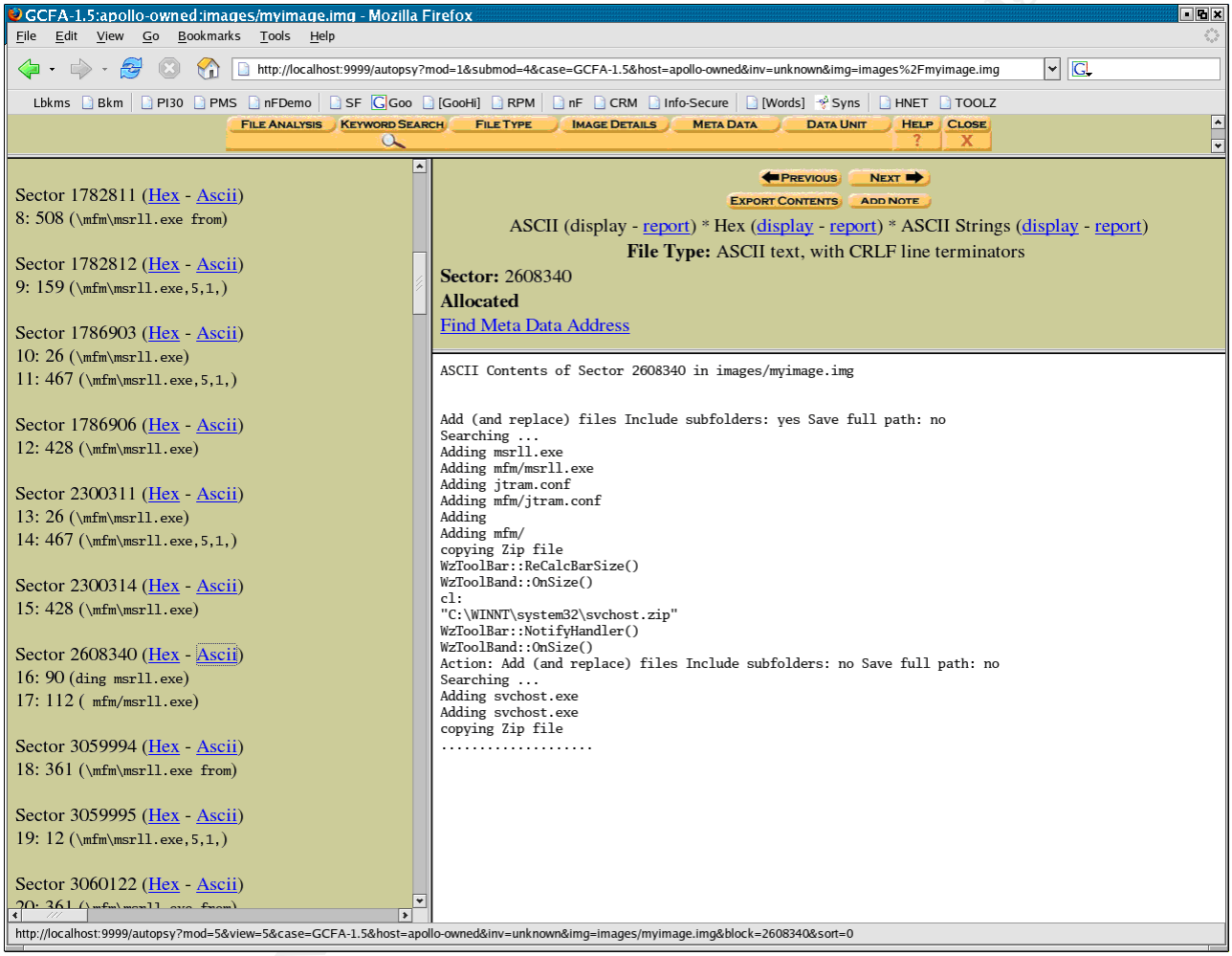
```
Thu Jul 01 2004 11:38:30      0 m.. -/-rwxrwxrwx 0      0       28413785 C:\WINNT/inf/_ET4B2.tmp (deleted)
Thu Jul 01 2004 11:38:30      0 m.. -/-rwxrwxrwx 0      0       31370303 C:\WINNT/HELP/_ET4B0.tmp (deleted)
```

They indicate the last use of the system right before imaging.

To conclude, tracing timelines involves parsing massive amounts of data manually. We realize that one can always say something more from the timeline (even on FAT file system where some time stamps are less than useful). As we stated before, we focused our investigation on malware and related activities.

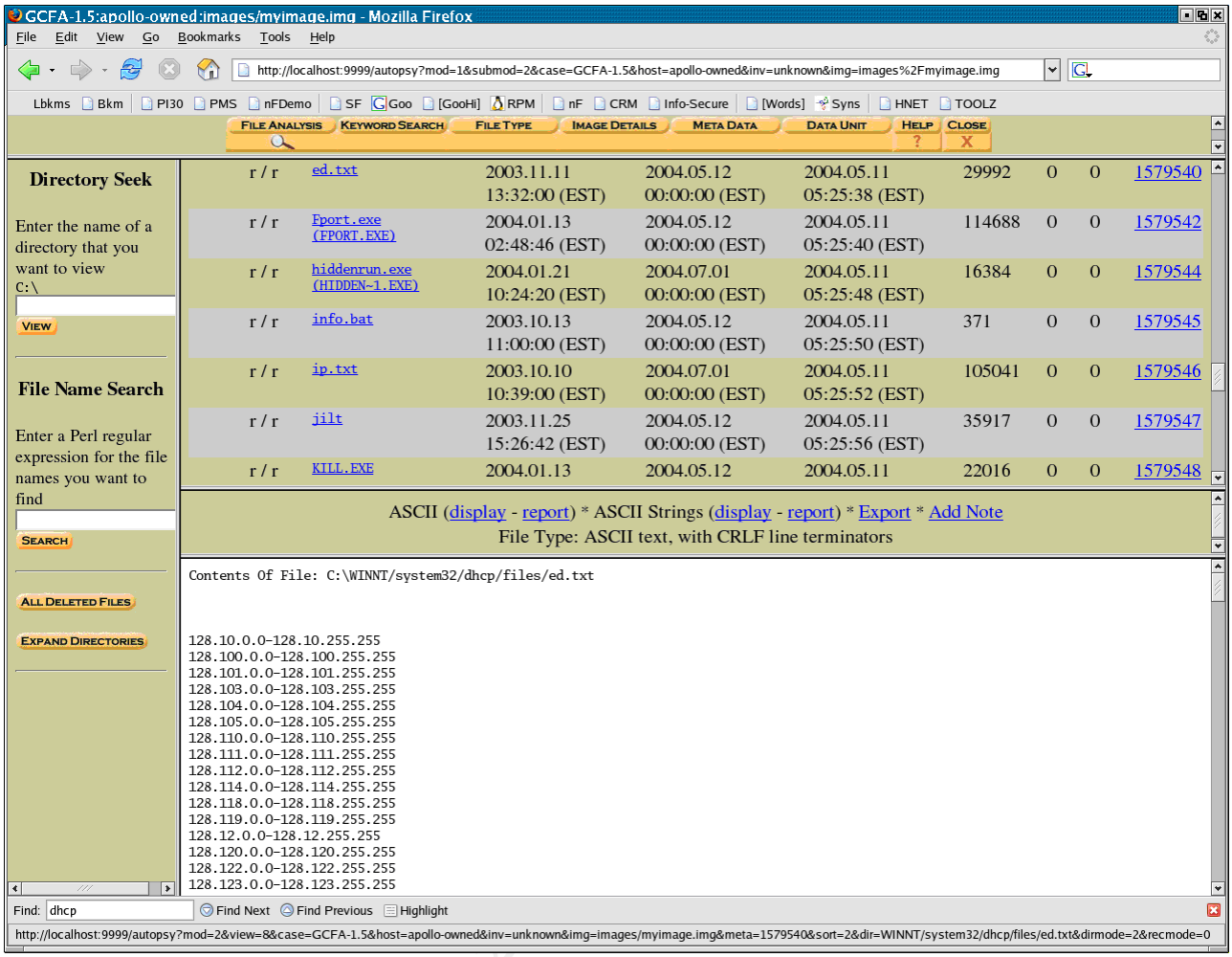
String Search:

We looked for other copies of the same Trojan as well as other strings with Trojan names (possible consequences of executing Trojans). Examples include “msrll.exe”, “mdll.exe” and “jtram.conf”. We used Autopsy/Sleuthkit to search for strings in the image.



For example, the above image shows one hit when searching for the “msrll” string – the name of the Trojan. The image shows the log of the WinZip archiver (obviously used by the administrator team to package the Trojan for investigation!) packaging the files. There are too many hits for the Trojan names since they also show up in anti-virus and Windows Event logs (the latter due to anti-virus logging to the event log).

Searching for the string “mdll.exe” uncovered a treasure trove of malware and hacker tools in *C:\WINNT\system32\dhcp\files.* Here are some of the examples:



Shown are the "hiddenrun.exe" executable wrapper, "Fport.exe" port scanner[29], text files containing configuration information for hiding tools and IRC communication (shown below) and bat files with various commands to be run on the systems, such as the "update.bat" file.

What is also interesting is that in the above directory is that contains the erased copies of the above software dated 05/10/2004 and then fresh copies (unerased) from 05/13/2004. That indicated reinfection after manual cleaning without fixing the core problem – an obvious security lapse.

Here is the full list of strings we looked for:

String	Reason for search	Discovery summary
msrll.exe	Trojan name, search for other occurrences	Multiple references discovered in various logs, file found in one location

[29] Likely a version from Foundstone.com site

msrll	Trojan name, search for other occurrences	Multiple references discovered, file found in one location
dhcp/files	Location of suspicious files, search for other occurrences	Location contains a large number of suspicious files
hiddenrun	Name of the suspicious binary file, search for other occurrences	Discovered in registry and filesystem
Fport	Name of the suspicious binary file, search for other occurrences	Multiple copies discovered, likely from multiple infections

We also performed keywords searches within the registry using AccessData Registry Viewer to look for the Trojan names in the registry. We found no others beyond what is shown above and involves the Run registry key.

Another quick and simple way to do string searches is to extract all the strings from the image and the search for strings in the resulting file. This technique can be applies in this case:

First, we build a list of all strings:

$ strings myimage.img > myimage.img.str

In our case, the file containing strings is very large (>1 GB) and so we compress it to save disk space

$ bzip2 *myimage.img.str*

which results in a much smaller 300MB compressed file *myimage.img.str.bz2*

Now we can look for strings in the *compressed* file, decompressing it on the fly (we sacrifice search time, but gain disk space):

$ bzcat myimage.img.str.bz2 | grep –i msrll.exe > msrll_exe-strings- myimage.img.str

The above command decompresses the file on the fly (through *bcat2*), looks for a string (through *grep*) results (in our case) and dumps the data into a file.

Here is what the file contains (excerpt):

716190984 220601010704,20,2,0,APOLLO,Administrator,,,,,,,16777216,"Unable to restore C:\WINNT\system32\mfm\msrll.exe from backup file after clean failed.",0,,0,,,,,0,,,,,,,,,,,,,,,,,,,,8.1.825
716191165
220601010704,5,1,0,APOLLO,Administrator,Backdoor.IRC.Bot,C:\WINNT\system32\mfm\msrll.exe,5,1,4,256,33570852,"",1088658024,,0

```
,,89128961,40195,0,0,0,0,0,0,20040625.019,32535,2,4,89128960,,{831F39E1-38E1-43A9-91B5-7D26B0419591},,,,,,8.1.825
716256520 220601010704,20,2,0,APOLLO,Administrator,,,,,,,16777216,"Unable to restore C:\WINNT\system32\mfm\msrll.exe from
backup file after clean failed.",0,,0,,,,,0,,,,,,,,,,,,,,,,,,8.1.825
716256701
220601010704,5,1,0,APOLLO,Administrator,Backdoor.IRC.Bot,C:\WINNT\system32\mfm\msrll.exe,5,1,4,256,33570852,"",1088658024,,0
,,89128961,40195,0,0,0,0,0,0,20040625.019,32535,2,4,89128960,,{831F39E1-38E1-43A9-91B5-7D26B0419591},,,,,,8.1.825
3559674967  848 msrll.exe
3559698296  848 msrll.exe
```

The above indicates that the string '*msrll'* was indeed found on this machine in various contexts (antivirus logs, directory entries, etc). The disadvantage of this string search method is that one cannot quickly pinpoint the location on the disk where the string was found.

However, we can quickly search for strings without scanning through the whole image, which has a chance of being humongous. Autopsy can also search by using the string file, like we did manually above.

In case of a malware investigation, string searches are less important compared to insider abuse, copyright or content piracy, since most of the interesting things are located in known malicious files or files that are pinpointed by the antivirus solutions or manual file system review[30]. However, we managed to file the fun directory of hacker tool based on string search for the Trojan name.

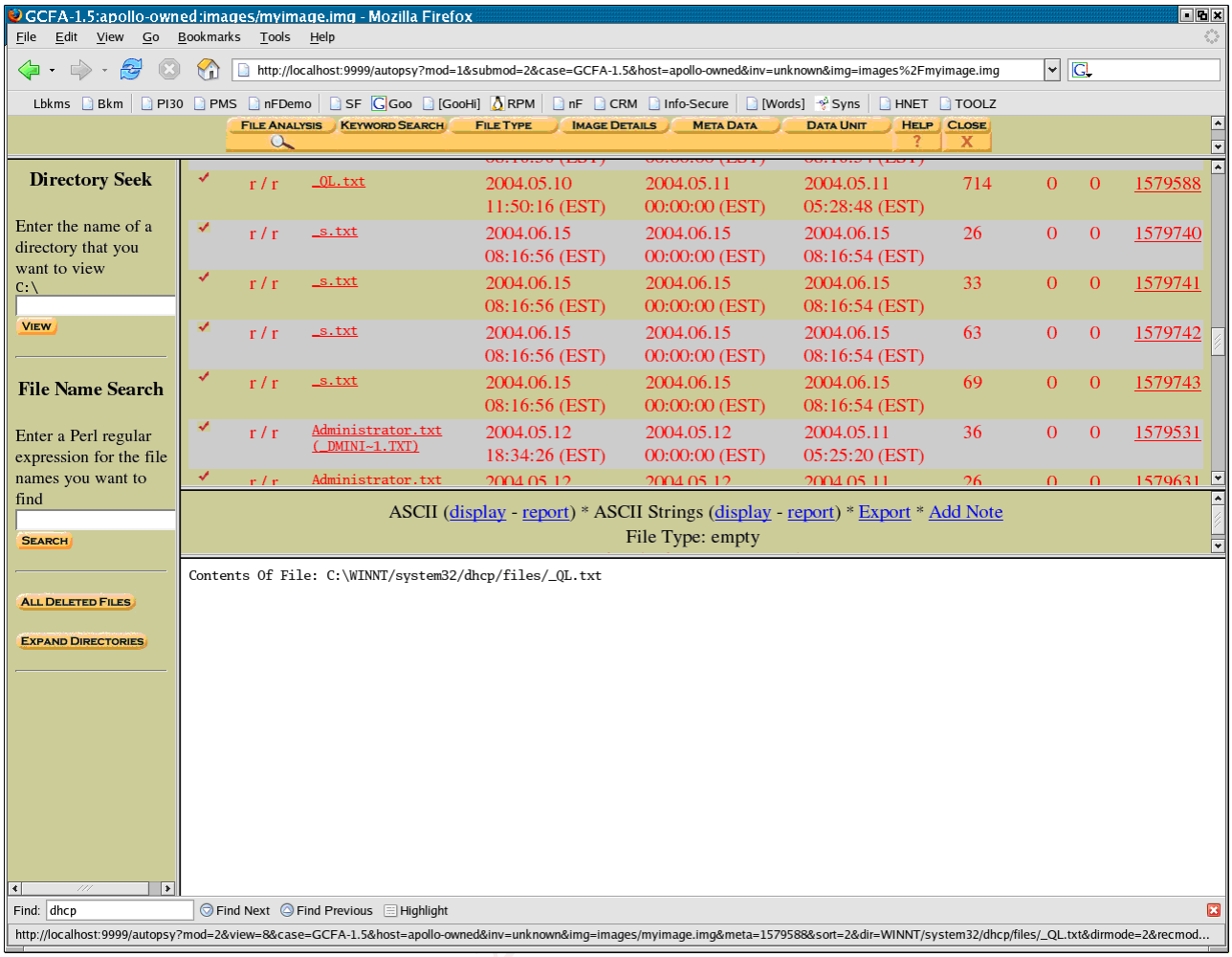
Recover Deleted Files:

Deleted file are important for this investigation, since we need to determine whether the initial recovery by the system administrator has caused any damage. Also, we wanted to follow the reinfection trail. Autopsy/Autopsy allows easy undeletion of files across all platforms (including FAT that we have in this case).

In this case, we have attempted to recover some of the deleted attack tools from the *C:\WINNT\system32\dhcp\files* referenced in the above section. We wanted to compare the erased toolset from the currently deployed one to better understand the reinfection that was occurring.

We have tried to recover the files using Autopsy/Autopsy as shown below:

[30] This applies to less advanced "stock" malware and not to custom-written automated penetration agents



However, such recovery failed for unknown reasons. We hypothesize that since the files are dated 05/10/2004, they might have been overwritten and thus no content remained on disk.

In this case, we decided to go the “hard way” and fish for chunks of the files using a forensics tool “foremost”[31]. This tool uses binary signatures to look for specific file types within the bulk of data such as the disk image.

Here is how we did it:

1. Download the latest version of ‘foremost’ to our Linux analysis systems from http://foremost.sourceforge.net/
2. Unarchive it

$ tar zxf foremost-0.69.tgz

3. Compile it

$ cd foremost-0.69

$ make

[31] Available at http://foremost.sourceforge.net/

4. Become root
$ su
<enter root password>
5. Install the tool
 # make install

In our undeletion test, we first separated the unused space (where the deleted files are) of the image from the full image so that our "foremost" run will not simply copy the *existing* files from the image, but will only look for deleted files.

This was accomplished by running "dls" tool from the 'sleuthkit'.

$ dls –f fat16 myimage.img > myimage.img.deleted

The above command copies the unallocated space to a separate file (flag '–f' indicated a file system type, which was FAT16 in this case[32]).

We then modified the default *foremost* configuration alone, so that the tool will look for files of the following types:

1. Images (*.jpg, *.gif, *.png)
2. HTML document (*.html, *.htm)
3. Movies files (*.mpg)
4. Zip archives (*.zip)

All of the above files do have characteristic traits in their headers or other sections.

Specifically, here is what the configuration file contained:

Type	Case sensitive?	Size	Header binary signature	Footer binary signature
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
png	y	200000	\x50\x4e\x47?	\xff\xfc\xfd\xfe
mpg	y	4000000	\x00\x00\x01\xba	\x00\x00\x01\xb9
mpg	y	4000000	\x00\x00\x01\xb3	\x00\x00\x01\xb7
htm	n	50000	<html	</html>
zip	y	10000000	PK\x03\x04	\x3c\xac

We can then run foremost like this:

$ foremost -v -q -c foremost.conf -o /home/anton/0_TODO/GCFA/new-evidence/recovered/ myimage.img.deleted

[32] Unlike FAT12 for a floppy in Part I

The command options stand for:

-v verbose output with more details
-q quick mode (only look for header signatures in the beginning of the cluster – usually reliable enough and always much faster)
-c location of a configuration file (described above)
-o output to the directory named *recovered*

Here is what happens when the above command runs. Foremost tries to match the bits for the beginning configuration file with the strings within the image. If a match is found, it will either look for the match related to the end of file or simply will extract a pre-configured number of bytes (as set in the configuration file).

The results of our run recovered the following files that appeared in the *recovered* direcory:

```
01/28/2005  01:09 PM         1,206 00000000.gif
01/28/2005  01:09 PM       200,000 00000001.png
01/28/2005  01:10 PM         4,368 00000003.htm
01/28/2005  01:10 PM         2,290 00000005.htm
01/28/2005  01:10 PM            64 00000006.htm
```

The GIF image 00000000.gif did open successfully and looked like this:

The HTML files also were visible and turned out to be pieces of HTML email (not shown here for confidentiality reasons)

The PNG image however, turned out to be a “false positive”. The file did not contain any image data , but plain text with the string PNG in the first line (which seem to have confused the pattern matching code of the *foremost*). The file has a size of 200000 bytes which means that *foremost* did not match the footer code but extracted the maximum size.

Thus, file recovery with foremost is quick and easy, *provided* that the files can be defined in terms of string signatures and/or sizes.

Conclusions:

We conclude the following:

- the system in question **was infected** by multiple malware specimen (see section on string search and antivirus log analysis)
- a known **backdoor malware** application was installed and running on

the system (see section on string search and antivirus log analysis); other malware was also detected
- judging from the networking activity and host data, the **attacker planned to use this system** for expanding his presence on the network (due to discovered 'fport.exe' scanner) as well as IRC communication. IRC was also likely used for remote control (due to observed periodic polling of the IRC servers from the system)
- the system had the **latest service pack** possible for Windows 2000 – Service Pack 4.
- infection **did not happen through the client** (Internet Explorer), but rather through direct scanning (see IRE history file analysis)
- antivirus software on the system was **ineffective** due to multiple reasons (see the analysis antivirus and Windows logs)
    - antivirus scan failures due to low disk space
    - antivirus update failures due to low disk space
    - inherent properties of the antivirus solutions trying to detect and clean backdoor applications and bots (which are not viruses and behave differently)
- the system did **not have system monitoring** or log analysis software that would have warned about the partition overflow; it also did not run any other endpoint security solution (supported by the IRCR data on running services)

The system was actually rebuilt and returned to production before the investigation was concluded.

The following recommendations were formulated for this system to be more secure:

Top 3

1. Enable automated Windows updates and monitor their progress or use a patch management tool
2. Enable antivirus updates and monitor their progress or
3. Deploy a host-based (i.e. personal firewall) or at least enable Windows firewall

Top 10

1. Deploy and periodically perform spyware checks using freeware tools such as Ad-Aware or Spybot S&D or their commercial counterparts
2. Tighten down Internet Explorer settings or upgrade to an alternate browser to decrease the risk of web-borne malware
3. Increase system monitoring (host event logs) and centralize collection and analysis of logs
4. Setup more effective log reviews processes to facilitate intrusion and infection discovery

5. Train system admins in better recognizing intrusion traces and escalating to an incident response team
6. Distribute forensics evidence collection tools and train system admins in their effective use in case of an incident, as well as in proper chain of custody
7. Audit the outbound firewall policy to prevent Trojan communication

References

1. Tools
    a. "Forensic Acquisition Utilities" by George M. Garner http://users.erols.com/gmgarner/forensics/
    b. IRCR Toolkit by John McLeod http://www.securityfocus.com/tools/2024 and http://ircr.tripod.com/
2. Background on malware
    a. "Backdoor.IRC.Bot" write-up by Symantec Security Response http://securityresponse.symantec.com/avcenter/venc/data/backdoor.irc.bot.html (similar)
    b. "Troj/Tometa" write-up by Sophos http://www.sophos.com/virusinfo/analyses/trojtometaa.html (similar)
    c. "BKDR_JTRAM.A" write-up by Trend http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=BKDR_JTRAM.A (similar)
    d. http://www.mcse.ms/message597388.html
3. Related incidents
    a. Windows IT Pro Forums on Security http://www.winntmag.com/Forums/messageview.cfm?catid=42&threadid=123027 (July 02, 2004)
    b. What is "msrll.exe"? http://www.winntmag.com/Forums/messageview.cfm?catid=42&threadid=123027 (March 20, 2004)
    c. "run-time error at startup" http://www.mcse.ms/message597388.html (April 20, 2004)
    d. "Compromised FTP Server Investigation" by Anton Chuvakin http://www.linuxsecurity.com/feature_stories/ftp-analysis-part1.html

Appendix A Password Tried for Data Recovery (Part I)

Passwords tried:

<Empty>
Robert
robert
ROBERT
"Robert"
John

john
JOHN
Robert J
Robert J.

Appendix B Strings Found within the DLL File (Part I)

HTML file information:

```
<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>
```

Remained of the content:

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

Appendix C Glaring Security Lapses – IRCR Data (Part II)

This sheds some light as to why the system in Part II was infected/compromised. This is also provided as a part of "lessons learned" for the community.

1. Account policy ("what not to do")

```
                Incident Response Collection Report (IRCR)

 Computer Name: OWNEDBOX
 Domain Name: WORKGROUP
 Time/Date: 16:27:26 Fri Jul 02  2004 Eastern Daylight Time

 -------------------------------------------------------------------------------
 net accounts - list displays the current settings for password, logon
 limitations, and domain information.
 -------------------------------------------------------------------------------

 Force user logoff how long after time expires?:       Never
 Minimum password age (days):                          0
 Maximum password age (days):                          42
 Minimum password length:                              0
 Length of password history maintained:                None
 Lockout threshold:                                    Never
 Lockout duration (minutes):                           30
 Lockout observation window (minutes):                 30
 Computer role:                                        SERVER
 The command completed successfully.
```

3. Running services ("what not to run")

```
Incident Response Collection Report (IRCR)

 Computer Name: APOLLO
 Domain Name: WORKGROUP
 Time/Date: 16:27:29 Fri Jul 02  2004 Eastern Daylight Time

-------------------------------------------------------------------------------
 net start - displays a list of running services.
-------------------------------------------------------------------------------

These Windows 2000 services are started:
   Alerter
   Automatic Updates
   Background Intelligent Transfer Service
   COM+ Event System
   DefWatch
   DHCP Client
   Distributed Link Tracking Client
   Distributed Transaction Coordinator
   dllhost
   DNS
   DNS Client
   Event Log
   firewall
   FTP Publishing Service
   Gateway Service for NetWare
   IIS Admin Service
   License Logging Service
   Logical Disk Manager
   Messenger
   Network Connections
   NT LM Security Support Provider
   Plug and Play
   Print Spooler
   Protected Storage
   Remote Access Connection Manager
   Remote Procedure Call (RPC)
   Removable Storage
   Rll enhanced drive
   RunAs Service
   Security Accounts Manager
   Simple Mail Transport Protocol (SMTP)
   Symantec AntiVirus Client
   syslock
   System Event Notification
   Task Scheduler
   TCP/IP NetBIOS Helper Service
   Telephony
   Telnet
   Terminal Services
   Windows Management Instrumentation
   Windows Management Instrumentation Driver Extensions
   Workstation
   World Wide Web Publishing Service

The command completed successfully.
```

4. Listening daemons (“backdoor bonanza”)

Incident Response Collection Report (IRCR)

Computer Name: APOLLO
Domain Name: WORKGROUP
Time/Date: 16:27:31 Fri Jul 02 2004 Eastern Daylight Time

---

netstat - protocol statistics and current TCP/IP connections

Check for odd ports listening for connections from other hosts.

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:21	0.0.0.0:0	LISTENING
TCP	0.0.0.0:25	0.0.0.0:0	LISTENING
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:113	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1026	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1030	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1085	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1087	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1090	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1097	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1212	0.0.0.0:0	LISTENING
TCP	0.0.0.0:2200	0.0.0.0:0	LISTENING
TCP	0.0.0.0:2212	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3372	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7800	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7801	0.0.0.0:0	LISTENING
TCP	0.0.0.0:46682	0.0.0.0:0	LISTENING
TCP	127.0.0.1:43958	0.0.0.0:0	LISTENING