



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

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

Analysis of an Unknown Floppy Image and a Hacked Server

GIAC Certified Forensic Analyst (GCFA)

Practical Assignment

Version 1.5 (April 30, 2004)

Author: Nihar S. Khedekar

Submitted on: 1st November 2004

Abstract

Robert Leszczynski, a designer at Ballard Industries attempted to take a floppy outside the R&D Labs of the company. A forensic examination of this floppy has to be performed to find its capabilities, content, purpose and possible use. First part of the paper describes the situation and complete analysis in detail, in a view to understand the concepts of Forensic analysis and develop a forensic report.

Draupadi, a Linux server, used as a web-server and a mail server was compromised in the month of August, 2004. The attack was identified only when a *rootkit* installed was easily detected in normal day to day activity. The system was immediately put out of service and all data transferred to another machine.

The second part of the paper describes the forensic analysis performed on the hard disk image of this machine. The aim of the analysis was to dig out the procedure that led to the compromise of the system.

The paper was written as an assignment for the Track-8 GIAC Certified Forensic Analyst training program conducted by GIAC. Basic understanding about the usage(commands, piping, redirection etc) of bash and GNU/Linux is expected from the reader.

© SANS Institute 2000 - 2005, Author retains full rights.

TABLE OF CONTENTS

<u>Abstract</u>	2
<u>Introduction</u>	4
<u>Legend</u>	4
<u>Tools Used</u>	4
<u>PART I</u>	6
<u>Case description</u>	6
<u>Examination Details</u>	6
<u>Time-Line Analysis</u>	17
<u>Image Details</u>	19
<u>Forensic Details</u>	21
<u>Program Identification</u>	21
<u>Legal Implications</u>	23
<u>Additional Information</u>	24
<u>PART II</u>	25
<u>FORENSIC INVESTIGATION OF A HACKED SERVER</u>	25
<u>Synopsis of the Case Facts</u>	25
<u>Verification</u>	25
<u>System Description</u>	26
<u>Hardware</u>	26
<u>Image Media</u>	26
<u>Media Analysis</u>	28
<u>Timeline Analysis</u>	45
<u>String Search</u>	47
<u>Conclusions</u>	47
<u>References</u>	49
<u>APPENDIX A</u>	50
<u>File-System Basics</u>	50
<u>The Slueth Kit</u>	50
<u>Command used to generate MAC Times</u>	51
<u>Linux commands</u>	52
<u>APPENDIX B</u>	53
<u>APPENDIX C</u>	54

Introduction

Forensics is not just analysis of data but its a process of gathering evidence, preserving it, analyzing it and generating a detailed report on the findings. Each investigation and analysis is unique in its own way. To make sure that nothing is missed out during the analysis, a flow is defined that helps to follow a systematic approach to the problem at hand.

In a broad sense the forensic analysis goes the following phases.

1. Information gathering. This part covers aspects related to evidence collection, integrity verification and maintaining the chain of custody
2. Analysis. This part describes the actual analysis in detail addressing the issues related to media analysis, time-line analysis without breaking the chain of custody and recording each and every action of the analysis.
 - (a) Media Analysis: Analysis of the found image.
 - (b) Time line analysis: Analysis of the timestamps found on all the files in that image.
3. Inferences. All that the analysis opened up.
4. Report. The most important aspect of a forensic analysis is reporting the findings.

The machine used for forensic analysis was a Acer Veriton 7600G, loaded with GNU/Linux-2.6.5-1.358smp, Fedora Core II. The hardware configuration is as follows:

512 MB RAM, Dual processor, Intel(R) Pentium(R) 4 CPU 2.80GHz, with 512KB cache, clocked at 2793.845MHz each, 81964 MB HDD

Legend

All content shown in Courier is the copy-paste of the command run on the console, along with its output. All command names followed by a parenthesis, enclosing a section name indicate that the information about this command was obtained from that section of the manual page documentation available with the GNU/Linux distribution of Fedora Core II. A brief introduction of all commands used is available in Appendix A.

eg. md5sum(1); Section 1 of the *man* page.

Tools Used

The complete examination was performed using The Slueth Kit(TSK) and Autopsy Forensic Browser. As per the website, both are open source digital forensics tools from Brian Carrier that run on Unix systems. TSK contains a set of utilities that can be used to read file-system images. A brief easy to understand explanation of each utility is available in the Appendix A. If you are naive to Forensics and File Systems, its worth spending a few moments reading Appendix A. The Autopsy Browser provides a web-based interface to all these TSK utilities.

There were also other Linux based utilities that were used, dcfldd(1), nc(1), strings(1), grep(1), mount(8), md5sum(1), ls(1), gunzip(1), head(1), tail(1), cat(1), diff(1), file(1), KHexEdit and certainly bash(1). A brief introduction to these utilities has been added in Appendix A. Process Explorer and Task Manager were used on Windows.

And finally, although it may not be called a tool, something that certainly played an important role in the analysis and findings was www.google.com, a search engine.

© SANS Institute 2000 - 2005, Author retains full rights.

PART I

ANALYSIS OF AN UNKNOWN FLOPPY IMAGE

Case description¹

Robert John Leszczynski, Jr., is employed by Ballard Industries, a designer of fuel cell batteries which produces specialized batteries used around the world by thousands of companies. Robert is assigned as the lead process control engineer for the project.

After several successful years of manufacturing and distributing a relatively new fuel cell battery, which is used in many applications, Ballard industries notices that many of their clients are no longer re-ordering from them.

After making several calls the vice president of sales determines 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 investigation has not turned up very much. It is apparent that Rift, Inc. somehow has received proprietary information from Ballard industries. Ballard industries keeps a customer database of all its clients and it is feared that that information somehow got out along with other proprietary data.

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.

This part of the paper contains analysis of the floppy disk and provides a report to David Keen, the security administrator.

Examination Details

The image was downloaded from http://www.giac.org/gcfa/v1_5.gz and decompressed using `gunzip(1)`. Following was the information regarding the image that was provided on the website.

Tag# fl-260404-RJL1

3.5 inch TDK floppy disk

MD5: d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img

fl-260404-RJL1.img.gz

```
[root@agdrd19 downloads]# gunzip -t v1_5.gz
```

```
[root@agdrd19 downloads]# gunzip -N v1_5.gz
```

```
[root@agdrd19 downloads]#
```

No errors were detected while decompressing the downloaded file. `gunzip(1)` was also run with the `-t` option to perform a integrity test of the compressed file. The `-N` option ensured that the original file name and time stamp is restored

¹ http://www.giac.org/GCFA_assignment.php

if present.

A cryptographic hash was obtained using md5sum(1) and checked with the one mentioned on the website information. It was found to be matching, and so it can be confirmed that the download was proper and verifies the integrity of the image. This hash will now be considered as a baseline signature for all future integrity checks.

```
[root@agdrd19 sans]# md5sum fl-260404-RJL1.img
d7641eb4da871d980adbe4d371eda2ad fl-260404-RJL1.img
[root@agdrd19 sans]#
```

The above output also confirms that the chain of custody has not been broken. The file(1) command was run on the downloaded image to learn more information about it. The command attempts to determine the file type, and perform three tests, viz FileSystem test, magic number test and language test, in that order. More details about find(1) can be obtained from the manual page.

```
[root@agdrd19 sans]# file fl-260404-RJL1.img
fl-260404-RJL1.img: x86 boot sector, code offset 0x3c, OEM-ID " mkdosfs", root
entries 224, sectors 2872 (volumes <=32 MB) , sectors/FAT 9, serial number
0x408bed14, label: "RJL      ", FAT (12 bit)
[root@agdrd19 sans]#
```

From the output its clear that the floppy was formatted with a FAT12 file system, with 9 clusters used to take up one FA Table.

To know further details about the file-system and the data on the image, the command 'fsstat' was run. This utility is a part of The Slueth Kit.

```
[root@agdrd19 sans]# fsstat -f fat12 fl-260404-RJL1.img
FILE SYSTEM INFORMATION
```

File System Type: FAT

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

Sectors before file system: 0

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

META-DATA INFORMATION

Range: 2 - 45426

Root Directory: 2

CONTENT-DATA INFORMATION

Sector Size: 512

Cluster Size: 512

Total Cluster Range: 2 - 2840

FAT CONTENTS (in sectors)

105-187 (83) -> EOF

188-250 (63) -> EOF

251-316 (66) -> EOF

317-918 (602) -> EOF

919-1340 (422) -> EOF

1341-1384 (44) -> EOF

[root@agdrd19 sans]#

The information doesn't exhibit any abnormalities. However the command has helped us know some important information related to the image.

1. Each cluster is of 512 bytes, which is the same as the sector size. This indicates that each cluster will correspond to just one sector.
2. We have a total of 2872 clusters, ie total space of $2872 * 512 = 1470464$ (1436 Kb) This indicates that the floppy has 4Kb of waste space. This could be because of bad sectors. A normal floppy has 2880 Clusters (1440Kb)

Now that the integrity had been verified and the filesystem known, the image was mounted using the loopback option provided with mount(8). To make sure that the analysis does not modify the image, the image was mounted with the following options:

ro: READ ONLY

noatime: DONT PERMIT ANY CHANGES IN ACCESS TIME STAMP

noexec: DENY ANY EXECUTION OF BINARIES

nodev: IGNORE ANY DEVICE FILES(character/block special)

The first two options were used to make sure that the image was not tampered in any case. The option *noexec* was used to avoid any accidental execution of binaries on the image, that could spoil the forensic system itself. And the *nodev* option to avoid usage of any device files of at all present on the image.

```
[root@agdrd19 sans]# mount fl-260404-RJL1.img floppy-image/ -o  
ro,loop,noatime,noexec,nodev
```

```
[root@agdrd19 sans]# ls floppy-image/ -s  
total 640
```

```
-rwxr-xr-x 1 root root 22528 Apr 23 2004 Acceptable_Encryption_Policy.doc  
-rwxr-xr-x 1 root root 42496 Apr 23 2004 Information_Sensitivity_Policy.doc
```

```
-rwxr-xr-x 1 root root 32256 Apr 22 2004 Internal_Lab_Security_Policy1.doc
-rwxr-xr-x 1 root root 33423 Apr 22 2004 Internal_Lab_Security_Policy.doc
-rwxr-xr-x 1 root root 307935 Apr 23 2004 Password_Policy.doc
-rwxr-xr-x 1 root root 215895 Apr 23 2004 Remote_Access_Policy.doc
[root@agdrd19 sans]#
```

After a successful mount, a simple `ls(1)` showed that the files, "Password_Policy.doc" and "Remote_Access_Policy.doc" are larger than the other files(10 times!). This was noted, for further analysis. Also two files "Internal_Lab_Security_Policy1.doc" and "Internal_Lab_Security_Policy.doc" were observed to differ in size.

The next command to be run was ``fls'`, to list all the available as well as deleted files in the image. It showed an interesting result:

```
[root@agdrd19 sans]# fls -f fat12 fl-260404-RJL1.img
r/r 3:  RJL      (Volume Label Entry)
r/r * 5:      CamShell.dll (_AMSHLL.DLL)
r/r 9:  Information_Sensitivity_Policy.doc (INFORM~1.DOC)
r/r 13: Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
r/r 17: Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
r/r 20: Password_Policy.doc (PASSWO~1.DOC)
r/r 23: Remote_Access_Policy.doc (REMOTE~1.DOC)
r/r 27: Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
r/r * 28:      _ndex.htm
[root@agdrd19 sans]#
```

It shows that two files, CamShell.dll and Index.htm were deleted. To find out what were the contents of these files, the command ``dls'` was run on the image to analyze the deleted data of the file.

```
[root@agdrd19 sans]# dls -f fat12 fl-260404-RJL1.img | less
<HTML>
<HEAD>
<meta http-equiv=Content-Type content="text/html; charset=ISO-8859-1">
<TITLE>Ballard</TITLE>
</HEAD>
<BODY bgcolor="#EDED"ED">

<center>
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflas
h.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"
PLUGINSOURCE="http://www.macromedia.com/go/getflashplayer"></EMBED>
```

```
</OBJECT>
</center>
</BODY>
</HTML>
```

--- Followed by some binary data ---

It showed an HTML header of some deleted file. The HTML text showed a file named "ballard.swf". This string was searched in the image using grep(1) and the offset was obtained.

```
[root@agdrd19 sans]# grep ballard.swf fl-260404-RJL1.img -ba
17265: <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=""
[root@agdrd19 sans]#
```

To find the file's inode number(using `ifind`) the block number would be necessary. The block number was obtained as follows:

BlockNumber = offset / BlockSize

=> BlockNumber = 17265 / 512

=> BlockNumber = 33

The `ifind` gave an interesting result. It showed two inodes for the same block.

```
[root@agdrd19 sans]# ifind -f fat12 fl-260404-RJL1.img -ad 33
5
28
```

```
[root@agdrd19 sans]#
```

`icat` could be used to find out what these deleted files actually contained. But before that it would be interesting to know more information about these inodes.

The command `istat` was run on the image to find out detailed information of the inodes. The timezone of the incident location is MST which demands a skew of 45000² seconds. Since FAT12 doesn't store last accessed *time*, but only the date, the change shown by skew should be neglected.

```
[root@agdrd19 sans]# istat -s 45000 -f fat12 fl-260404-RJL1.img 5
```

Directory Entry: 5

Not Allocated

File Attributes: File, Archive

Size: 36864

Num of links: 0

Name: _AMSHHELL.DLL

Adjusted Directory Entry Times:

Written: Sat Feb 3 07:14:16 2001

Accessed: Sun Apr 25 11:30:00 2004

Created: Sun Apr 25 21:16:18 2004

Directory Entry Times:

² MST = -700 => 25200 secs; IST(Indian Std Time) = +5:30 => 19800 secs; 45000 = 25200 + 19800.

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

Sectors:
33

Recovery:

33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48

49 50 51 52 53 54 55 56

57 58 59 60 61 62 63 64

65 66 67 68 69 70 71 72

73 74 75 76 77 78 79 80

81 82 83 84 85 86 87 88

89 90 91 92 93 94 95 96

97 98 99 100 101 102 103 104

[root@agdrd19 sans]# istat -s 45000-f fat12 fl-260404-RJL1.img 28

Directory Entry: 28

Not Allocated

File Attributes: File, Archive

Size: 727

Num of links: 0

Name: _ndex.htm

Adjusted Directory Entry Times:

Written: Thu Apr 22 22:23:56 2004

Accessed: Sun Apr 25 11:30:00 2004

Created: Sun Apr 25 21:17:36 2004

Directory Entry Times:

Written: Fri Apr 23 10:53:56 2004

Accessed: Mon Apr 26 00:00:00 2004

Created: Mon Apr 26 09:47:36 2004

Sectors:
33

Recovery:

33 34

[root@agdrd19 sans]#

The output showed that there were two DELETED files. This is confirmed because `istat` showed the inodes as `Not Allocated`.

`ffind` was used to find out the filenames of these inodes.

[root@agdrd19 sans]# ffind -af fat12 fl-260404-RJL1.img 28

* /_ndex.htm

[root@agdrd19 sans]# ffind -af fat12 fl-260404-RJL1.img 5

* /CamShell.dll (_AMSHLL.DLL)

[root@agdrd19 sans]#

The information given by istat also indicates that the file `CamShell.dll` may not be recovered completely. This is because the first two sectors have been overwritten by the file `index.htm`. Moreover six sectors following the first two contain ZEROes. This could be some deliberate attempt to destroy evidence.

Both the files were recovered using the `icat` tool available.

```
[root@agdrd19 sans]# icat -r -f fat12 fl-260404-RJL1.img 28 | tee
analysis/recovered/index.htm
```

```
<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/swflas
h.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"
```

```
PLUGINS PAGE="http://www.macromedia.com/go/getflashplayer"></EMBED>
```

```
</OBJECT>
```

```
</center>
```

```
</BODY>
```

```
</HTML>
```

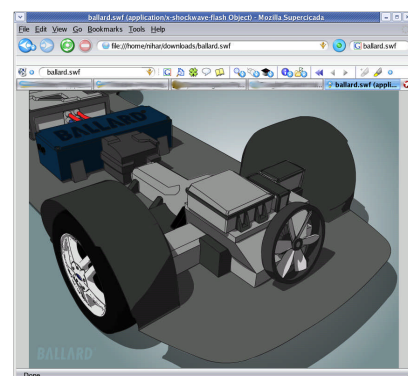
```
[root@agdrd19 sans]# icat -r -f fat12 fl-260404-RJL1.img 28 >
```

```
analysis/recovered/CamShell.dll
```

```
[root@agdrd19 sans]#
```

INDEX.HTM

Although *index.htm* was recovered completely, its not yet clear what was it used for. Apart from embedding a Flash/Shockwave object, named, *ballard.swf* the file doesn't seem to cause any suspicion. There is infact a file *ballard.swf* available on the web[10].



CAMSHELL.DLL

The second deleted file is certainly a cause for concern. CamShell.dll is a windows based dynamically linked library that is used by a program called Camouflage. Camouflage is a tool used for steganography. Its a windows based

tool, that can hide one file of any type into another(called *carrier*) without causing any difference in the normal behaviour of the carrier file. The embedded file can be conveniently extracted from the carrier, again using Camouflage. Moreover Camouflage is also capable of scrambling the data before embedding it into the file. Also it can protect the extraction with a password.

CamShell.dll was assumed to be a Dynamically Linked library based on its file extension. However its important to confirm this. Since the head part of the file was overwritten with index.htm, the *tail* end of CamShell.dll was of primary importance. The software, Camouflage was downloaded from the web. Its freely available for download at (Freeware License)³

Some web-blogs⁴ mentioned Camouflage being used to distributed MP3s and to contain Backdoor/Virus/Trojan.

This required utmost care while installation and usage. However the blogs also mentioned that the Trojans are deployed only when the MP3 are used. This could very well be true, and at the same time could also mean that Camouflage as a software was harmless but was improperly used by those embedding mp3 files using it. This was exactly what was quoted on the webpage.

The program was installed using normal user privileges. Process Explorer was used to monitor file operations of the program. At the end of installation, a time stamp based query was run to find all the files that were modified in the system by the installation. Nothing suspicious was found. The installation was assumed to be clean.

To confirm the validity of CamShell.dll as a Dynamically Linked Library, the file had to be compared with the newly downloaded one. But since the CamShell.dll recovered from the floppy was corrupted, that corrupted part was replaces using same part from the newly downloaded file to generate *NewCamShell.dll*. All this was done using tail(1), head(1), diff(1) and cat(1).

```
[root@agdrd19 sans]# file analysis/recovered/CamShell.dll
analysis/recovered/CamShell.dll: HTML document text
[root@agdrd19 sans]# file downloaded/CamShell.dll
downloaded/CamShell.dll: MS-DOS executable (EXE), OS/2 or MS Windows
[root@agdrd19 sans]# head -c 4096 downloaded/CamShell.dll >
analysis/NewCamShell.dll
[root@agdrd19 sans]# du -b analysis/recovered/CamShell.dll
36864  analysis/recovered/CamShell.dll
[root@agdrd19 sans]# tail -c `echo "36864 - 4096" | bc`
analysis/recovered/CamShell.dll >> analysis/NewCamShell.dll
[root@agdrd19 sans]# file analysis/NewCamShell.dll
analysis/NewCamShell.dll: MS-DOS executable (EXE), OS/2 or MS Windows
[root@agdrd19 sans]# diff -s analysis/NewCamShell.dll
downloaded/CamShell.dll
Files analysis/NewCamShell.dll and downloaded/CamShell.dll are identical
[root@agdrd19 sans]#
```

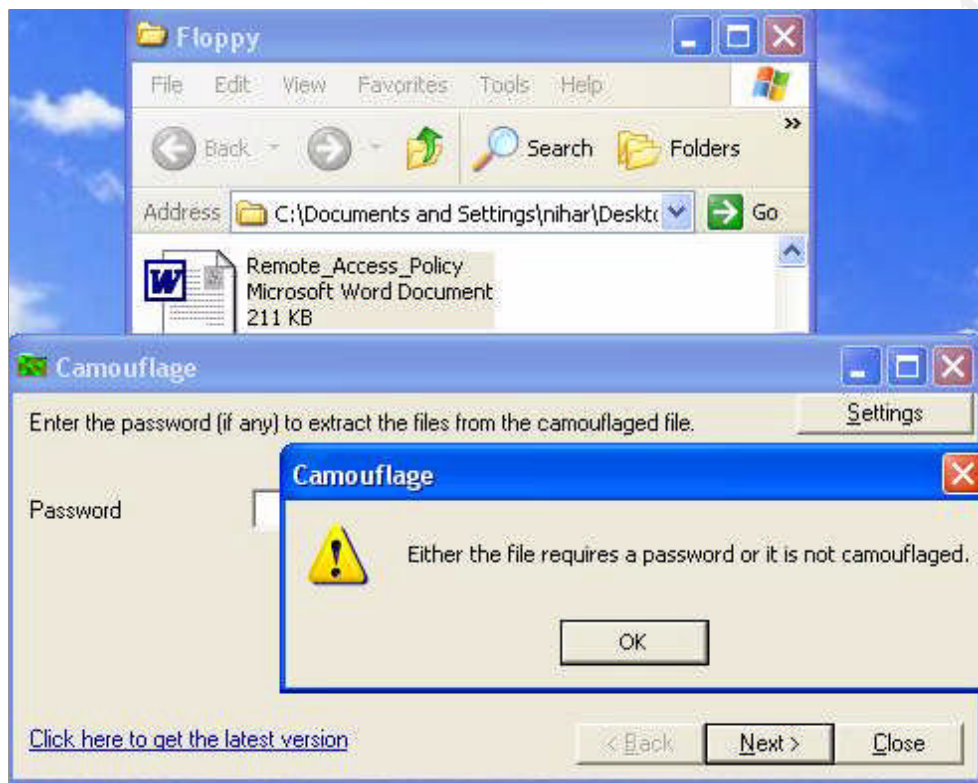
The above test considered sufficient to guarantee that the file CamShell.dll

³ <http://camouflage.unfiction.com/>

⁴ <http://www.tranceaddict.com/forums/archive/topic/79627-1.html>

recovered from the floppy was indeed a Dynamically Linked Library that resided on the floppy some moment in time.

The presence of CamShell.dll and abnormal file sizes, pointed to the floppy containing some hidden data. Camouflage was run on the files that were larger in size compared to others viz "Password_Policy.doc" and "Remote_Access_Policy.doc". The results weren't heartening. The software said that there was a password used whiled hiding, or there may not be any files embedded at all!



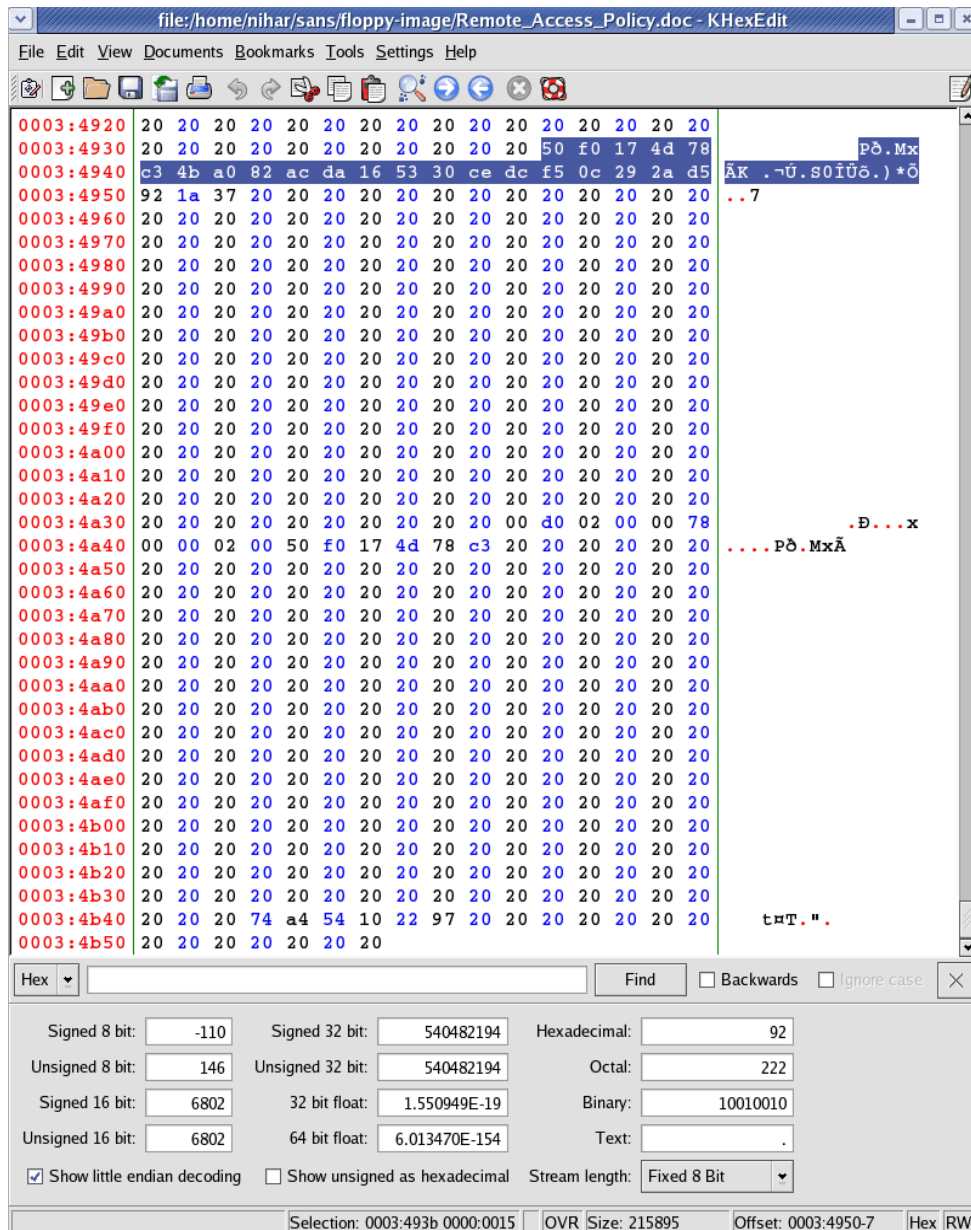
But, file-sizes being 10 times the other similar files, certainly gave an indication that these files had to be steganographed. A quick strings(1) on the image gave out something interesting - "pwReserved". It was immediately tried out on the files, but it didn't work! Apart from this, there was not anything much suspicious or something demanding attention.

The next approach was to look for cracking Camouflage's password. A google search for 'detect crack camouflage password' resulted in a pdf⁵ document on explaining steganographic methods and various detection techniques. The slides also pointed to another link⁶ that explained the password recovery mechanism for files steganographed using Camouflage. Apparently, Camouflage stored the passwords in the carrier itself. Its very interesting and relatively simple technique to recover the password. The technique has been explained in Appendix B. It was a very exciting read and has showed an approach to password cracking. Its also worthwhile to note that there are softwares on the web that are widely used despite of their incompetence.

⁵ <http://www.blackhat.com/presentations/>

⁶ <http://www.guillermi2.net/stegano/camouflage>

The password cracking mechanism was attempted to recover the password from the file "Remote_Access_Policy.doc".



```

50 F0 17 4D 78 C3
XOR 02 95 7A 22 0C A6
= 52 65 6D 6F 74 65

```

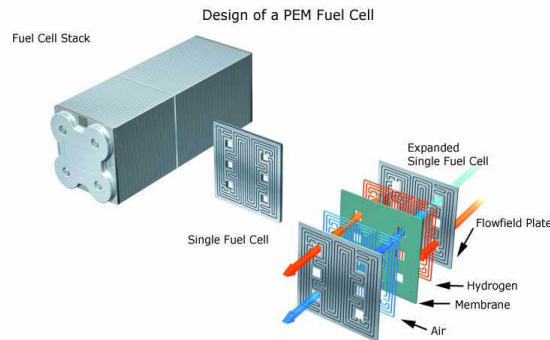
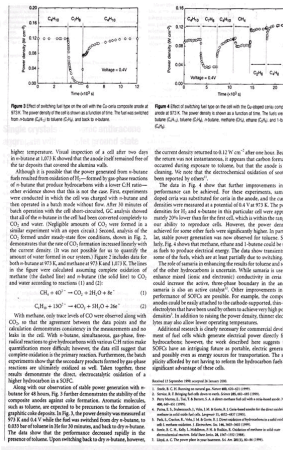
```

[root@agdrd19 sans]# echo -e "\x52\x65\x6D\x6F\x74\x65"
Remote
[root@agdrd19 sans]#

```

This password did work on "Remote_Access_Policy.doc", and Camouflage unwrapped a file *CAT.mdb* hidden in the document. *CAT.mdb* was a file containing a small database of client information, apparently of Ballard Industries.

Looking at the relationship between the password and filename, the password for the file "Password_Policy.doc" was guessed as 'Password'. Some images referring to the design of *Proton Exchange Membrane Fuel Cell* were found hidden within the document. The following files were recovered.

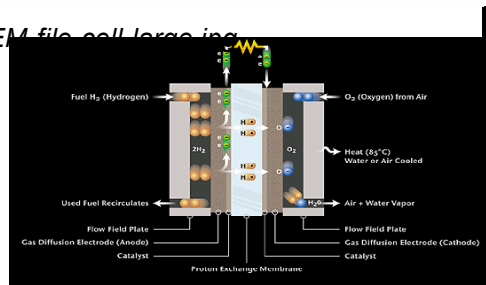


1. 3Hydrocarbon fuel cell page2.jpg

1. Hydrocarbon%20fuel%20cell%20page2.jpg

2. PEM-fuel-cell-large.jpg

3. pem_fuelcell.gif



1. 2pem_fuecell.gif

Next the software opened up "Internal_Lab_Security_Policy.doc". The result was good to look at. Camouflage detected an embedded file named, "Opportunity.txt". The file contained interesting information.

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

The contents clearly indicated that Leszczynski did indulge into unfair activities. The second last line - *They are available as we discussed - "First Name"* - seemed to have some hidden message. The phrase "*First Name*" must have been a hint for the recipients to guess passwords.

With this analysis the following inferences can be made.

1. Mr. Leszczynski, tried to leak important information that seems proprietary and confidential of Ballard Industries, with an unfair intention to gain monetary benefits. Irrespective of the technical information being proprietary or not, the Client Database is very important for Ballard Industries, and the consequences of this information leak could be devastating, as Ballard Industries could lose its invaluable clientele, and finally loose market.
2. The *latest schematics* that Leszczynski attempted to leak, though seem to be

of the new design, may not be considered proprietary until anyone qualified by Ballard Industries guarantees so. Moreover the image *pem_fuelcell.gif* is publicly available on the web[9].

3. Although he made an attempt to leak information, the seizure of floppy indicates that the attempt was not successful. However the very attempt by Mr. Leszczynski to smuggle information, indicates that he is in touch with some people willing to pay for this information. Mr. Leszczynski can prove to be an important link to reach such people.

Tools like *sorter* and *lazarus*⁷ were run on the image. But there was nothing much suspicious about the content. Nothing more than what was already found was revealed. Following is the output generated by *lazarus*:

```
T....Hh.....!!!!!!Tttt.....Tttt.....Tttt.....Tt..Tttt.....XxxxAaaaaaaaAaaaaaTt..Tttt.....Xxxx  
xXxxxxxxTt..Tt.....
```

The presence of sound media`!' arouses a doubt that it could be the shockwave/flash file that we were looking for, but a quick look at the location shows that the media file is located at sector 30, which according to FAT12 is the root directory! The another presence of media is at sector 47, which is a part of the deleted file "CamShell.dll". Again this possibility is ruled out because, a md5sum check with the downloaded CamShell.dll showed that the files were same.

Time-Line Analysis

The Autopsy Forensic Browser (Version: 2.01) was used to generate time-lines for the floppy's image. The time line generated can be found as a separate file along with this submission.

When files are copied from a machine to the floppy, the modification time stamp of the file remains whatever was before copying. Since an access is made during copy, the access time stamp changes. The created times stamp shows when the file was created on the floppy. An examination of the time-line has led to the following conclusions:

1. The files were copied from the machine to the floppy on Apr 25th 2004 between, Sun Apr 25 2004 21:16:18 and Sun Apr 25 2004 21:17:44.
2. There were no changes made in the files after they were copied on the floppy.
3. The floppy seems to have been formatted on the Sat Apr 24 2004 22:23:40 This was inferred from the time-line entry:
Sat Apr 24 2004 22:23:40 0 m.c-/rwxrwxrwx 003 RJL (Volume Label Entry)
4. The time-stamps also indicate that Mr. Leszczynski did all this activities after office hours.
5. The times indicate that all files except index.htm were copied together; ie Selected together, and then placed on floppy. This is because of very minor difference in their creation timestamps. There is a delay of about minute after that last of these files(Acceptable_Encryption_Policy.doc) was created and

⁷ <http://www.lazarus.org/>

index.htm. This indicates that index.htm was copied separately.

5. The file, index.htm has a later creation time stamp than CamShell.dll. This, and inference in point 5 could mean that CamShell.dll was deleted, and then index.htm was copied on the floppy. This could have caused index.htm being written on the sectors earlier occupied by CamShell.dll before being deleted. The fact that FAT12 searches for free sectors from the start also supports the above theory. This seems to be the reasons why the first sector of CamShell.dll contained html text.

© SANS Institute 2000 - 2005, Author retains full rights.

Image Details

1. List of all files in the image: The following list generated using `fls` show two deleted files.

```
[root@agdrd19 sans]# fls -f fat12 fl-260404-RJL1.img
r/r 3:  RJL      (Volume Label Entry)
r/r * 5:  CamShell.dll (_AMSHHELL.DLL)
r/r 9:  Information_Sensitivity_Policy.doc (INFORM~1.DOC)
r/r 13: Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
r/r 17: Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
r/r 20: Password_Policy.doc (PASSWO~1.DOC)
r/r 23: Remote_Access_Policy.doc (REMOTE~1.DOC)
r/r 27: Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
r/r * 28:  _ndex.htm
[root@agdrd19 sans]#
```

2. True name of the program used: Camouflage.

3. File/MAC Time information from the image and their **sizes** (bytes): The local timezone being IST, a skew of 45000 seconds is needed to obtain time stamps in MST. The following command(Explanation available in Appendix A) gave time-stamps of all the files as well as their size(shaded) in bytes. Time stamps are Modified, Accessed and Change in that order.

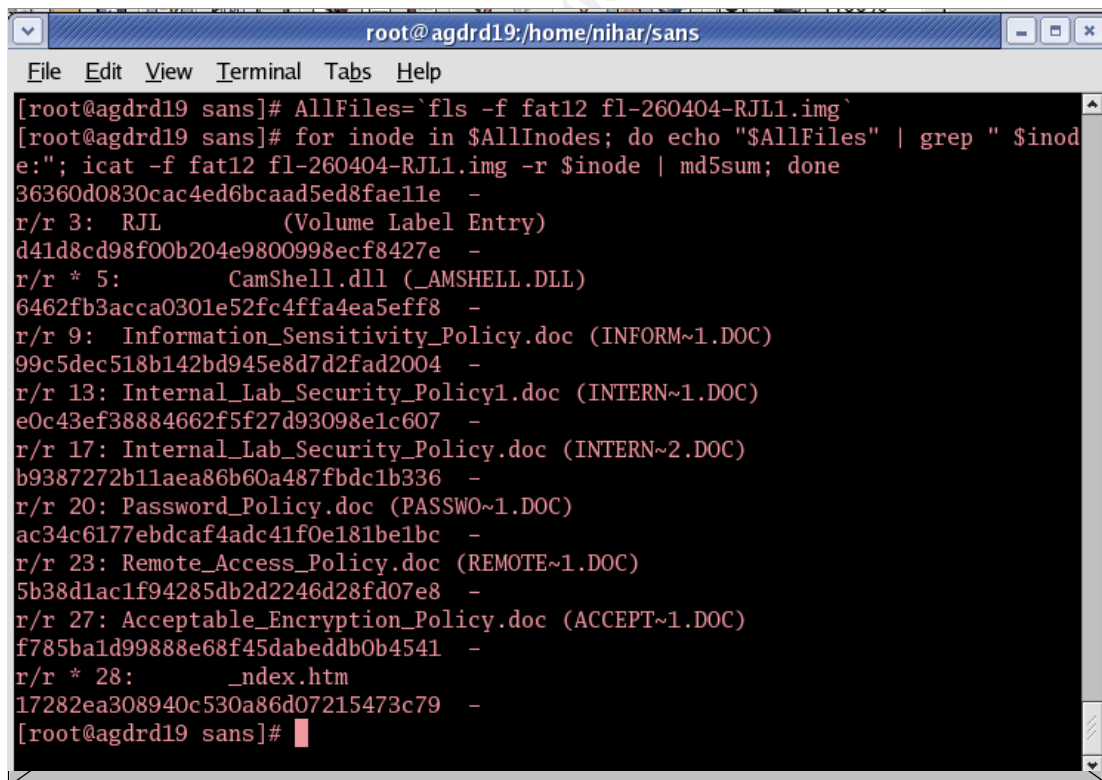
```
[root@agdrd19 sans]# fls -f fat12 -l -zMST -s 45000 fl-260404-RJL1.img | sed
's/)    /\n  /g'
r/r 3:  RJL      (Volume Label Entry)
      2004.04.24 22:23:40 (MST)
      2004.04.24 00:00:00 (MST)
      2004.04.24 22:23:40 (MST)
      0      0      0
r/r * 5:  CamShell.dll (_AMSHHELL.DLL)
      2001.02.03 07:14:16 (MST)
      2004.04.25 00:00:00 (MST)
      2004.04.25 21:16:18 (MST)
      36864  0      0
r/r 9:  Information_Sensitivity_Policy.doc (INFORM~1.DOC)
      2004.04.23 01:41:10 (MST)
      2004.04.25 00:00:00 (MST)
      2004.04.25 21:16:20 (MST)
      42496  0      0
r/r 13: Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
      2004.04.22 04:01:06 (MST)
      2004.04.25 00:00:00 (MST)
      2004.04.25 21:16:22 (MST)
      32256  0      0
r/r 17: Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
      2004.04.22 04:01:06 (MST)
      2004.04.25 00:00:00 (MST)
      2004.04.25 21:16:24 (MST)
      33423  0      0
r/r 20: Password_Policy.doc (PASSWO~1.DOC)
```

```

2004.04.22 23:25:26 (MST)
2004.04.25 00:00:00 (MST)
2004.04.25 21:16:26 (MST)
307935 0 0
r/r 23: Remote_Access_Policy.doc (REMOTE~1.DOC)
2004.04.22 23:24:32 (MST)
2004.04.25 00:00:00 (MST)
2004.04.25 21:16:36 (MST)
215895 0 0
r/r 27: Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
2004.04.23 01:40:50 (MST)
2004.04.25 00:00:00 (MST)
2004.04.25 21:16:44 (MST)
22528 0 0
r/r * 28: _ndex.htm 2004.04.22 22:23:56 (MST)
2004.04.25 00:00:00 (MST)
2004.04.25 21:17:36 (MST)
727 0 0
[root@agdrd19 sans]#

```

4. File Owners: Since FAT12 doesn't support users, file owners cannot be specified.
5. MD5 Hashes. Again the same variable AllNodes was reused once again, but a new variable AllFiles was used. The following screen shot shows the filename followed by its md5 hash.



```

root@agdrd19:/home/nihar/sans
File Edit View Terminal Tabs Help
[root@agdrd19 sans]# AllFiles=`ls -f fat12 fl-260404-RJL1.img`
[root@agdrd19 sans]# for inode in $AllNodes; do echo "$AllFiles" | grep " $inode:"; icat -f fat12 fl-260404-RJL1.img -r $inode | md5sum; done
36360d0830cac4ed6bcaad5ed8fae11e -
r/r 3: RJL (Volume Label Entry)
d41d8cd98f00b204e9800998ecf8427e -
r/r * 5: CamShell.dll (_AMSHLL.DLL)
6462fb3acca0301e52fc4ffa4ea5eff8 -
r/r 9: Information_Sensitivity_Policy.doc (INFORM~1.DOC)
99c5dec518b142bd945e8d7d2fad2004 -
r/r 13: Internal_Lab_Security_Policy1.doc (INTERN~1.DOC)
e0c43ef38884662f5f27d93098e1c607 -
r/r 17: Internal_Lab_Security_Policy.doc (INTERN~2.DOC)
b9387272b11aea86b60a487fbdclb336 -
r/r 20: Password_Policy.doc (PASSWO~1.DOC)
ac34c6177ebdcaf4adc41f0e181be1bc -
r/r 23: Remote_Access_Policy.doc (REMOTE~1.DOC)
5b38d1ac1f94285db2d2246d28fd07e8 -
r/r 27: Acceptable_Encryption_Policy.doc (ACCEPT~1.DOC)
f785ba1d99888e68f45dabeddb0b4541 -
r/r * 28: _ndex.htm
17282ea308940c530a86d07215473c79 -
[root@agdrd19 sans]#

```

5. Keywords. Dirty Word List: Ballard, Rift, money, secrets, confidential, battery, batteries , fuel cell, design, competitors, clients, proprietary.

Forensic Details

The forensic analysis done so far has proved fruitful, and has resulted in the following conclusions:

1. Mr. Leszczynski tried to smuggle files outside the R&D Labs using the software Camouflage
2. Camouflage is a program used for steganography. The program is used to embed one file into another(called *carrier*), without affecting the normal operation of the carrier. Detailed discussion on Camouflage is available in the section "Examination Details" and Appendix B of this paper.
3. Although the file CamShell.dll has a later Accessed Time-Stamp(Sun Apr 25 2004), it has not been used on either of the files available on the floppy image. Thats because, none of the files have a modification time-stamp later than Apr 25 2004. However it could be very well possible that the file was used to generate a different file that might be residing on Mr. Leszczynski machine. Limiting the search to the Steganographed files found in the floppy, its concluded that the program Camouflage was last used on Thu Apr 22 2004 23:25:26, which is the latest modification time of one of the Steganographed files- Password_Policy.doc.

Program Identification

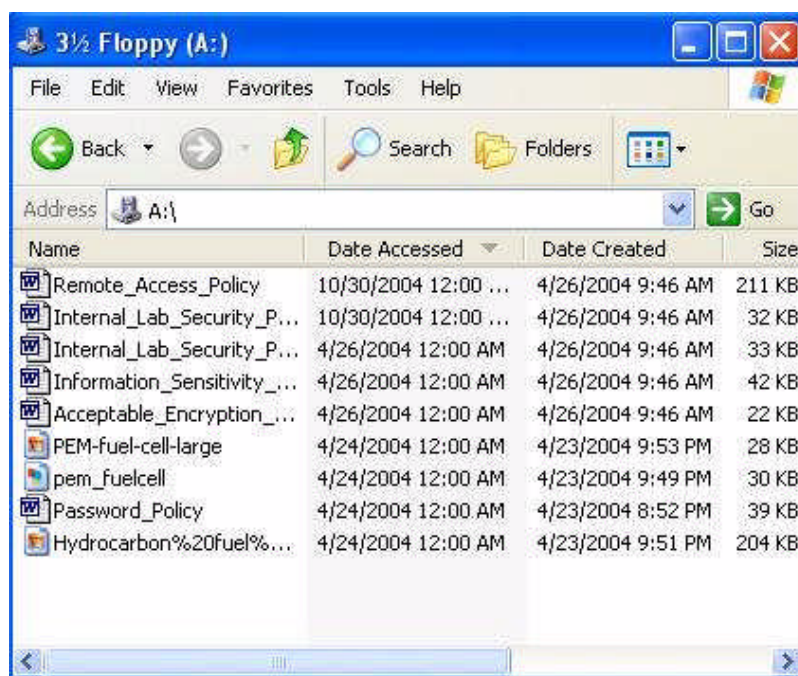
Earlier analysis could recover the hidden files that Mr. Leszczynski tried to smuggle outside the R&D Lab. However to prove that it was the same program that he used to embed the files, it was necessary that the same operation be redone, and shown to generate identical output. To do this, the extracted files were re-hid in the same document file. However the resultant carrier document generated did not match the ones found in the evidence. Multiple attempts to Camouflage files, showed that every time Camouflage generated a different output. Three files were created, Password_Policy.doc, Password_PolicyAE1.doc and Password_PolicyAE3.doc, all embedded with the same files – Hydrocarbon%20fuel%20cell%20page2.jpg, PEM-fuel-cell-large.jpg and pem_fuelcell.gif.

```
[root@agdrd19 reCamouflaged]# md5sum *
2663555ac1576b12b5e563ba2644c59e Password_PolicyAE1.doc
006e0398339cf6e5644fd6d5214b2ebd Password_PolicyAE3.doc
ac34c6177ebdc4f4adc41f0e181be1bc Password_Policy.doc
[root@agdrd19 reCamouflaged]#
```

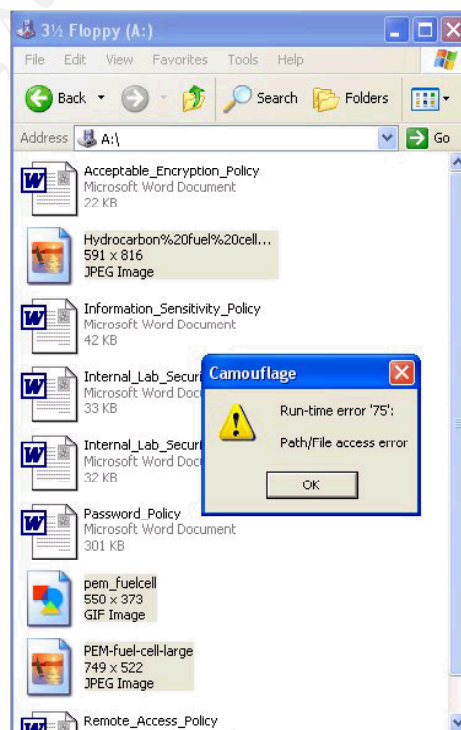
Camouflage when used, also stores the timestamps of the files that are being hid. This causes the new carrier file generated to have a different signature everytime.

This can be proved by extracting files from the evidence, and looking for the times stamps of the files embedded within.

To retain the time-stamps of the extracted files, they were directly extracted onto a floppy using Camouflage. This made sure that the File System data structures of the extracted files remained the same. The following screen shot shows it.



To make sure that the time stamps stay intact while embedding the files, the floppy was made Write-protected, by opening the Write-Protect window. The floppy was once again put back into the Windows machine and Camouflage was tried. Camouflage could not perform the operation and failed with an error (Shown in the adjacent screen shot). A simple right click on the files and selecting “Camouflage” failed with error! Perhaps Camouflage needs write access to the directory where it reads the data. This behavior of the software may not let us prove that it was the same program that Mr. Leszczynski use. Had it been possible to do something similar to mount's -o noatime in Windows, it would have been possible to verify the claims.



Legal Implications

Though it was not possible to prove that Camouflage was the same program that Robert Leszczynski used to hide the files, he still did try to smuggle information outside the R&D Lab.

However, until and unless it cannot be ascertained that the information Leszczynski attempted to leak is in fact confidential information (Minimal Sensitivity, More Sensitive or Most Sensitive), he may not be convicted of violating Information Sensitivity Policy of the company.

If the policy documents found in the floppy are to be believed, Robert Leszczynski, could be found violating the Information Sensitivity Policy. According to the Section 3.3; Policy-Most Sensitive of the Information Sensitivity Policy followed by Ballard Industries, Robert is liable to face **Penalty for deliberate or inadvertent disclosure**, which implies *up to and including termination, possible civil and/or criminal prosecution to the full extent of the law*.

But before Robert could be found guilty it is important to ascertain the documents to be confidential. If the documents retrieved hidden within the files in the floppy are declared as *“technical information integral to the success of the company”*, the content will be tagged *“Most Sensitive”*. According to the same document found in the floppy, its clearly mentioned in Section 3.3, “Most Sensitive” documents need not be explicitly marked so. As per section 2.0 of the same document it’s the responsibility of the user to use common sense judgment in securing Ballard Industries Confidential information to the proper extent. If an employee is uncertain of the sensitivity of a particular piece of information, he/she should contact their manager.

The technique used by Leszczynski to transfer documents outside the R&D Labs does not follow any of the procedures mentioned in the *electronic distribution* techniques specified by the document. This itself is a policy violation by Leszczynski, provided the policy documents found in the floppy are correct.

Finally, if Leszczynski is to be penalized for violating Ballard Industries’ Policy, as per section 4.0 of the Information Sensitivity Policy, he will be subject to disciplinary action, up to and including termination of employment.

According to the § 72⁸ of the THE INFORMATION TECHNOLOGY ACT, 2000 (No. 21 of 2000), MINISTRY OF LAW, JUSTICE AND COMPANY AFFAIRS, India, Robert Leszczynski is liable to Penalty for breach of confidentiality and privacy. Robert Leszczynski is liable to be punished for with imprisonment for a term, which may extend to two years, or with fine, which may extend to one lakh rupees (Rupees One Hundred Thousand), or with both.

⁸ http://www.mit.gov.in/itbillonline/it_frame.asp

Additional Information

Numerous references played an important role in the complete life cycle of the analysis.

- [1] Website dedicated to open-source software development for detection of Steganography: <http://www.outguess.org>.
- [2] A SANS paper, titled, *Steganography: The Ease of Camouflage* on the ease if usage of Camouflage, a windows based steganographic tool. <http://www.sans.org/rr/papers/20/762.pdf>
- [3] This is where the steganographic software, *Camouflage* was downloaded from. <http://camouflage.unfiction.com/>
- [4] Official website for TSK and Autopsy: <http://www.sleuthkit.org/>
- [5] The website that explained cracking the password for Camouflage. A good website in itself, it helped in learning password cracking approach too. <http://www.guillermi2.net/stegano/camouflage/index.html>, <http://www.guillermi2.net/stegano/>
- [6] An enlightening presentation on the concepts Steganography and Steganalysis. <http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-raggo/bh-us-04-raggo-up.pdf>
- [7] The HoneyNet *Challengers*, whose solutions were invaluable in the analysis: <http://www.honeynet.org/scans/scan24/sol/dennis/index.htm>, <http://www.honeynet.org/scans/scan26/sol/nick/>
- [8] The publicly available *pem_fuelcell.gif* image_ http://www.ballard.com/be_informed/media_resources/image_gallery/category/Technology
- [9] <http://www.ballard.com/resources/animations/animations/FuelCellShort/ballard.swf>; The link that showed where ballard.swf was located: <http://www.overgrow.com/edge/showthread/t-539698.html>
- [10] <http://www.rtems.com/ml/rtems-users/2003/july/msg00149.html>
- [11] Time zone codes and the offsets. <http://atm.geo.nsf.gov/ieis/time.html>
- [12] Case description in PART I http://www.giac.org/GCFA_assignment.php
- [13] The lazarus utility <http://www.lazarus.org>
- [14] http://www.mit.gov.in/itbillonline/it_framef.asp
- [15] And finally, <http://www.google.com>

PART II

FORENSIC INVESTIGATION OF A HACKED SERVER

Synopsis of the Case Facts

Software Techniques Ltd⁹. used Draupadi, a Linux box as a web server as well as mail server. The organization, based in Bangalore, India, had nearly a fifty employees working in their office at Bangalore. All the users had accounts on Draupadi, and used it for their regular mail interactions. Software Techniques Ltd. also used Draupadi to host its website. All the system administration was handled by a team of three, led by Mr. K. Arjuna.

On 5th of August 2004 a regular user, Karan, working after office hours, noticed a change in behavior of some programs. Simple utilities like ls and telnet were giving problems. Unfortunately no-one from the system administration was around at this time. When Karan called up Arjuna, Karan was instructed to disconnect the machine from the network. All this happened around 10 pm IST.

The next day, Arjuna performed a minimal incident handling procedure and got the system back into the network. Draupadi was still not accessible from the Internet though. Finally after a few days, Arjuna transferred all the data from Draupadi to another machine, and stopped using this Linux Box.

The system was kept down before being handed over to a team comprising of Me, Pramod Pawar and Vijay Kumar V K – all Track 8, GCFA participants – to perform forensic analysis. Due to some administrative constraints, Mr. Arjuna denied permanent custody of the machine. However, we were permitted to image the partitions of the system.

This paper is a compilation of the analysis that I performed on the system. Although the images were handed over to a team as such, the paper comprises of my personal analysis and findings.

Verification

Draupadi was kept down for a long period of time(8-10 weeks) after the incident. In such a scenario the incident has mainly been verified based on interviews. Further analysis has helped verifying the incident.

When Karan called up Arjuna late that night, it was evident to Arjuna that Draupadi was compromised, and a rootkit must have been deployed. On his return to office the next morning, during the examination of the system, Arjuna made the following observations:

1. There were some new users(ids: ravi, diva and ro) added to the system. All of them had their home directories in /home/. This was contrary to Software Technique's regular practice of creating user home directories in /home/<dept-name>/login-id, these users had their directories in /home/login-id.

⁹All names(except the names used by the attacker) and have been replaced by fictitious ones, to preserve privacy.

2. The ACLs of the router were empty! This was one of the most astonishing facts of the case. Unfortunately, the router didn't maintain any logs.
3. Few suspicious processes named "superwu" were running on the system.

System Description

Draupadi was a Linux box loaded on an Siemens machine running Redhat 7.1, with a 2.4.2-2 kernel. The machine was used as a web as well as mail server with around fifty users logging into it daily. Apart from checking mails, this server was also host to some minor software development activities by the staff.

Web content was shtml/html/php. No other server side scripting was supported. The web link was a 2MBPS leased line connectivity. The machine was protected from the Internet cloud by a Cisco 2600 series router, with very strict ACLs. No ports other than 80(http) and 25(smtp) were open for this machine.

Draupadi also ran ssh, telnet, sftp and ftp daemons. However the access was restricted to the private IP Addresses only.

Hardware

Draupadi had the following configuration:

Siemens PRIMERGY-400,

Intel Pentium II Processor, clocked at 396.826 Mhz.,
256 MB RAM.

4x4Gb SCSI HDD, FUJITSU MAB3045SC.

CDROM Drive: SIEMENS STM/L S1.

Tape Drive(Sequential access): SONY SDT-7000.

1.44M Floppy Drive(/dev/fd0).

No hardware was permitted to be acquired as evidence. Following the best evidence rule, images of the media were obtained, details in next section.

Image Media

Due to some administrative and privacy constraints, Arjuna denied permanent custody of the machine and restricted us from imaging few of the sensitive partitions. Following the best evidence rule, we retorted to obtaining images of hard-disk partitions using dcfldd(1) and nc(1).

Since Draupadi was out of network, we transferred images onto the forensic analysis machine using a cross-cable, and tagged the images. The following output shows the md5 hashes taken on Draupadi. We had to use Knoppix, because Helix wouldn't detect the SCSI hard disks! Since dcfldd wasn't available, it was obtained from the network.

```
root@1[draupadi-knoppix]# ./dcfldd bs=10M if=/dev/sda7 hashwindow=0  
hashlog=sda7.md5 | nc 172.16.xx.xx 20017 -w 5
```

A sequence of such commands were run on Draupadi to image it. Finally we had all the md5 hashes stored in files, named with the respective device name.

```
root@1[draupadi-knoppix]# grep . *md5  
sda1.md5:Total: 661a4f317ce620e2f49de820a5d04257
```

```
sda10.md5:Total: 6b7bbf152e11e6f346357dc42c838d89
sda5.md5:Total: 22b2939c417e2f0333bf41dde891ebbf
sda7.md5:Total: 56a125d04fa2ea3beb9c355921ef9bda
sda8.md5:Total: cba7fada45bcaa8d0402cdd7d484c10b
sda9.md5:Total: debf77cc75c0e48ceb1274f9160d3abc
sdb1.md5:Total: b2ec6a068f2c57495a9ad39f1223c60d
sdc1.md5:Total: fe3df9d054d76fef3d038d1d604256b
sdc6.md5:Total: cfa9ce8308700f2ebfdef2424445a3cc
root@1[draupadi-knoppix]#
```

To make sure that these hashes were not lost, we also transferred all this hashes to the forensic machine. All the transferred partitions were checked for correctness by calculating their md5 cryptographic hash.

```
[root@agdrd19 analysis]# md5sum images/*
661a4f317ce620e2f49de820a5d04257 sda1-dd
6b7bbf152e11e6f346357dc42c838d89 sda10-dd
22b2939c417e2f0333bf41dde891ebbf sda5-dd
56a125d04fa2ea3beb9c355921ef9bda sda7-dd
cba7fada45bcaa8d0402cdd7d484c10b sda8-dd
debf77cc75c0e48ceb1274f9160d3abc sda9-dd
b2ec6a068f2c57495a9ad39f1223c60d sdb1-dd
fe3df9d054d76fef3d038d1d604256b sdc1-dd
cfa9ce8308700f2ebfdef2424445a3cc sdc6-dd
[root@agdrd19 analysis]#
```

All these hashes were added into Autopsy while importing images. The adjacent screen shot shows all the images that were imported into Autopsy.

As shown in the next screen shot, Autopsy verified all the images to pass the cryptographic hash test.

CASE GALLERY		HOST GALLERY	HOST MANAGER
mount		name	
/		images/sda10-dd	details
/boot/		images/sda1-dd	details
/home/		images/sdb1-dd	details
/tmp/		images/sda9-dd	details
/usr/		images/sda5-dd	details
/usr/local/		images/sda8-dd	details
/var/		images/sdc1-dd	details
/var/www/		images/sdc6-dd	details
swap		images/sda7-dd	details

IMAGE DETAILS

Name: images/sda10-dd

Mounting Point: /

File System Type: linux-ext2

MD5: 6B7BBF152E11E6F346357DC42C838D89

Host Directory: /forensics/draupadi_hack/server/

© SANS Institute 2000 - 2005, Author retains full rights.

Media Analysis

According to all the information acquired from Karan, other users and Arjuna, it was evident that there had to be a rootkit installed on the machine. The first thing that came to my mind was to look out for modification and access time stamps.

However I changed my mind and thought of doing the *inode-number test* first(explained later). I prefer noting down what things come to my mind, generate a check list, and tick them one by one with the shortest, simplest, quickest and the most effective one first. Its rather amusing how simple commands always turn out to be the quick evidence collectors.

Keeping this in mind I prepared a plan for my analysis. I would go through these one at a time, noting things on my way and avoid going wayward, as I often do.

1. Mount the images and do a quick and simple find(1) and ls(1) tricks to look out for bits and pieces of evidence.
2. By the time find(1) is doing its job, examine the /etc/passwd file and find about the newly added users Arjuna told about.
3. Get back with Autopsy and then check out some important timelines(will be explained in detail in the next section dedicated only to Time-Line analysis).
4. Look out for all the deleted files in the suspicious user directories.

QUICK FIND(1) AND LS(1) TRICKS

All the image files were mounted at a convenient location "/root/analysis/images/testmounts" in the forensic analysis machine, from where I ran my favourite sequence of commands. Care was taken during mounting to make sure that the images stay intact. I did this by mounting with the options:

ro: READ ONLY

noatime: DONT PERMIT ANY CHANGES IN ACCESS TIME STAMP

noexec: DENY ANY EXECUTION OF BINARIES

nodelv: IGNORE ANY DEVICE FILES(character/block special)

The first two options were used to make sure that the image was not tampered in any case. The option *noexec* was used to avoid any accidental execution of binaries on the image, that could spoil the forensic system itself. And the *nodelv* option to avoid usage of any device files of at all present on the image.

**find -name '...' -exec ls -ld --quoting-style=locale '{}' \; >
/root/searches/dot-dot-dot &**

Aim: To look out for all the files that started with three dots! Hardly do users create files or directories with such names.

How? find(1) seems to find almost anything you ask it to. The -name option specifies what name to look for. I passed '...' indicating dot-dot-dot followed by anything. Then I asked find(1) to execute ls -l command on the file-path that it

finds. The -exec option specifies so. The two curly braces are special for find. During execution, these get replaced by the file-path. The -l option of ls asks ls to list details of the filenames passed and the option --quoting-style=locale makes sure that filenames like "... " are not missed out. And finally all the output was redirected to /root/searches/dot-dot-dot, with the search put into background. By putting it into background, I can proceed with my other searches. I can then go back at leisure and analyze the outputs generated.

Sample Output: Note the back-tick and single-quote surrounding the filename.

```
-rw----- 1 509 300 715 Apr 5 2004 `./sdb1-dd/shakuni/duryodhan/...'
```

find -name superwu > /root/searches/superwu &

Next one had to be a search for the suspicious process found running.

My next aim was to find out what all things did the new users deploy all around the system. No matter what I do later, this search output will certainly be helpful. To do a uid based search, I needed to find the uid from /etc/passwd. This is where Autopsy helped. A quick visit to /, etc, passwd showed the following file contents

```
ravi:x:50101:50101:Ravi-Shadanah:/home/ravi:/bin/bash
duryodhan:x:50102:321:Duryodhan:/home/shakuni/duryodhan:/bin/bash
diva:x:50103:321::/home/diva:/bin/bash
ro:x:50104:50104::/home/ro:/bin/bash
```

find -uid 50103 > /root/searches/uid-diva &

find -uid 50101 > /root/searches/uid-ravi &

find -uid 50104 > /root/searches/uid-ro &

These commands would search for all the files owned by the user diva, ravi and ro.

From the contents of /etc/passwd, it seems as if the attacker was cautious while adding a new account. She made sure that the account had all the information filled. After a while, she might have noticed that she has been gone undetected and must have become careless. This can be inferred because the next id added was added without a username. She must have simply run the command useradd(8). This gave me sufficient reason to find out WHEN this was done last. Autopsy again, for ease of use, straight forward, reliable and predictable behavior! Autopsy showed that the file /usr/sbin/useradd was last accessed on Aug, 5th 2004, at 10:19 am. This meant that the user ro was added the very day the attacker was detected.

Next on the list of small tricks was sequence of inode numbers using ls(1).

The check had to be performed in three phases: Binaries, Libraries and then configuration files. In the directory, "/root/analysis/images/testmounts" I prepared links to all the mount points.

```
[root@agdrd19 testmounts]# ls -lG | grep ^l
lrwxrwxrwx 1 8 Oct 17 21:43 boot -> sda1-dd/
lrwxrwxrwx 1 8 Oct 17 21:44 home -> sdb1-dd/
```

```
lrwxrwxrwx 1 9 Oct 17 21:43 slash -> sda10-dd/
lrwxrwxrwx 1 8 Oct 17 21:44 tmp -> sda9-dd/
lrwxrwxrwx 1 8 Oct 17 21:43 usr -> sda5-dd/
lrwxrwxrwx 1 8 Oct 17 21:44 usr-local -> sda8-dd/
lrwxrwxrwx 1 8 Oct 17 21:45 var -> sdc1-dd/
lrwxrwxrwx 1 8 Oct 17 21:46 var-www -> sdc6-dd/
```

```
[root@agdrd19 testmounts]#
```

To check for inodes out of sequence, the following command was run:

```
[root@agdrd19 bin]# pwd
/root/analysis/images/testmounts/usr/bin
```

[root@agdrd19 bin]# ls -i | sort -n | less

ls with the option -i lists the inodes before the filename. This output when given to sort with the option -n gets its sorted numerically. During installation, files are normally installed at the same time, and so get sequence of inodes. In case any of these files are newly created, the new file will get an inode number that is out of sequence. Sorting the files based on inode number lets us pick out files like these.

Output of the above command did not show any abnormalities in either of the following directories:

```
/root/analysis/images/testmounts/usr/bin,
/root/analysis/images/testmounts/usr/sbin,
/root/analysis/images/testmounts/slash/bin
/root/analysis/images/testmounts/slash/sbin
```

To look out for the setuid files, the following command was run:

find -perm +6000 -ls > /root/searches/setuid-files &

find with the -perm option searches for files that have atleast ONE of the bits set as specified by the number. As per the man page of chmod(1), the first digit selects the set user ID (4) and set group ID (2) and sticky (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

To look for setuid files, the first digit should look for setuid(4) and setgid(2) => 2+4 = 6. Since rest of the bits can be ignored, the rest three digits were left zero.

```
[root@agdrd19 testmounts]# find -perm +6000 -ls > /root/searches/setuid-files
[root@agdrd19 testmounts]#
```

Next test was to check the modification time stamps of the files in the same directories. To do that I executed a simple command:

```
[root@agdrd19 bin]# pwd
/root/analysis/images/testmounts/slash/bin
```

[root@agdrd19 bin]# stat * | grep Modify | grep 2004


```
[root@agdrd19 bin]# cd ../../usr/bin/
[root@agdrd19 bin]# pwd
/root/analysis/images/testmounts/usr/bin
```

#[root@agdrd19 bin]# stat * | grep Modify | grep 2004

```
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-08-06 00:55:56.000000000
Modify: 2004-08-09 04:24:40.000000000
Modify: 2004-07-26 05:07:12.000000000
Modify: 2004-08-05 09:51:00.000000000
Modify: 2004-08-09 04:24:40.000000000
[root@agdrd19 bin]# pwd
/root/analysis/images/testmounts/usr/bin
[root@agdrd19 bin]#
```

The command stat(1) shows statistics of files, including the timestamps. All other files in this directory had timestamps in 2001 and 2002. With so many files to search from, it was necessary to put a filter, and that's why I did a grep for 2004.

There were two files shown modified on 26th of July, and 9th of Aug. If I can see the time stamps of the suspicious users home directory, I could get an idea of the duration from when Draupadi compromised. There were files with the same creation times in the home directory of the user ravi – the first suspicious user added.

```
[root@agdrd19 Desktop]# pwd
/root/analysis/images/testmounts/home/ravi/Desktop
[root@agdrd19 Desktop]# stat Autostart
File: `Autostart' -> `../../kde/Autostart'
Size: 17      Blocks: 0      IO Block: 4096   symbolic link
Device: 702h/1794d  Inode: 441677  Links: 1
Access: (0777/lrwxrwxrwx)  Uid: (  0/   root)  Gid: (  0/   root)
Access: 2004-08-08 08:41:23.000000000
Modify: 2004-07-26 05:22:16.000000000
Change: 2004-07-26 05:22:16.000000000
[root@agdrd19 Desktop]#
```

To check which files were having such a weird modification timestamp, I executed the following command. The command egrep is an extended grep where you can have some complex regular expressions to search for. In the following command, egrep looks for the filename or modify time stamps. This output is again sought for 2004 and if found 1 line before to it is shown. This is what -B option does.

```
[root@agdrd19 bin]# stat * | egrep 'Modify: |File:' | grep '2004' -B1
```

```
File: `a2ps'
Modify: 2004-08-09 04:24:40.000000000
--
File: `fax2ps'
Modify: 2004-08-09 04:24:40.000000000
--
File: `grops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `ksymoops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `oldps'
Modify: 2004-08-09 04:24:40.000000000
--
File: `pbmtolps'
Modify: 2004-08-09 04:24:40.000000000
--
File: `pdftops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `pfbtops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `pnmtops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `procmail'
Modify: 2004-08-06 00:55:56.000000000
--
File: `pstops'
Modify: 2004-08-09 04:24:40.000000000
--
File: `rem'
Modify: 2004-07-26 05:07:12.000000000
--
File: `strings '
Modify: 2004-08-05 09:51:00.000000000
--
File: `tiff2ps'
Modify: 2004-08-09 04:24:40.000000000
[root@agdrd19 bin]#
```

There we go. The output has nicely caught two files: `rem' and `strings '. Thanks to the quoting style used by stat(1), we can see there is a space followed after strings. A web search for such a small word like rem would be futile, and so will be for strings. But before setting out to dig out what these binaries are doing here, and their intentions, I decided to find some more things elsewhere too.

This is the list of directories I decided to look for trouble.

1. /usr/bin
2. /usr/sbin/
3. /bin
4. /sbin
5. /usr/local/bin
6. /usr/local/sbin
7. /usr/lib
8. /lib
9. /usr/local/lib

Unfortunately, the Change time-stamps for all the files were in a sequence, starting with 7th of August. This was because Mr. Arjuna used a crude data transfer mechanism. The file transfer was done by creating a archive first and then transferring it using ftp.

Getting back to the checklist. /usr/bin and /bin, /sbin and /usr/sbin were clean. Here is the output: No output infact!

```
[root@agdrd19 bin]# cd ../../slash/bin/
[root@agdrd19 bin]# stat * | egrep 'Modify: |File:' | grep '2004' -B1
[root@agdrd19 bin]# cd ../../slash/sbin/
[root@agdrd19 sbin]# stat * | egrep 'Modify: |File:' | grep '2004' -B1
[root@agdrd19 sbin]# cd ../../usr/sbin/
[root@agdrd19 sbin]# stat * | egrep 'Modify: |File:' | grep '2004' -B1
```

Still clear. Next was /usr/local/bin

```
[root@agdrd19 bin]# pwd
/root/analysis/images/testmounts/usr-local/bin
[root@agdrd19 bin]# stat * | egrep 'Modify: |File:' | grep '2004' -B1
  File: `backup'
Modify: 2004-05-05 00:19:39.000000000
--
  File: `core'
Modify: 2004-08-08 01:30:14.000000000
  File: `icheckd'
Modify: 2004-01-08 03:59:06.000000000 -0500
--
  File: `sweep'
Modify: 2004-01-08 03:59:06.000000000 -0500
[root@agdrd19 bin]#
```

All files except the core file had *safe* time-stamps. A did a simple gdb core core to check what caused the core file, and it said it was because of the command `w'. This must have been Arjuna trying to do `w' that caused this. This prompted me to deploy another find!

```
[root@agdr19 testmounts]# find -name core > /root/searches/cores
[root@agdr19 testmounts]# pwd
/root/analysis/images/testmounts
[root@agdr19 testmounts]#
```

/usr/local/sbin was empty. There was something real interesting at /usr/lib. /usr/lib/libc was found to be modified. I had heard about glibc, but whats this libc? Again somemore real good information!

```
[root@agdrd19 lib]# stat * | egrep 'Modify: |File:.' | grep '2004' -B1
File: `libc'
Modify: 2004-08-05 09:51:00.000000000
[root@agdrd19 lib]#
/lib also proved to be interesting!
[root@agdrd19 lib]# stat * | egrep 'Modify: |File:.' | grep '2004' -B1
File: `security'
Modify: 2004-08-05 09:50:51.000000000
[root@agdrd19 lib]# pwd
/root/analysis/images/testmounts/slash/lib
[root@agdrd19 lib]#
```

Next came /usr/local/lib. Although a little suspicious, but clear nevertheless.

```
[root@agdrd19 lib]# pwd
/root/analysis/images/testmounts/usr-local/lib
[root@agdrd19 lib]# stat * | egrep 'Modify: |File:.' | grep '2004' -B1 | sed 's/-
0400//g'
File: `libsavi.so.2' -> `libsavi.so.3.2.07.040'
Modify: 2004-01-08 03:59:10.000000000 -0500
File: `libsavi.so.3' -> `libsavi.so.3.2.07.040'
Modify: 2004-01-08 03:59:06.000000000 -0500
File: `libsavi.so.3.2.07.040'
Modify: 2004-01-08 03:59:06.000000000 -0500
[root@agdrd19 lib]#
```

So the final checklist for investigation:

```
"/usr/lib/libc"
"/bin/rem"
"/bin/strings"
"/usr/local/bin/core"
```

To probe more into the details of the binaries I tested them first with file(1) and then using strings(1). rem turned out to be a log remover! Seems like the attacker was all set to take over the machine! Here is the output of the

```
[root@agdrd19 bin]# strings rem
<skipped>
close
GLIBC_2.0
PTRh$
Usage: %s account
Log Cleaner by ^Sang^^pRaBu^, Copyright(c) 2002
#1 - Check log...
/var/log/wtmp
Can't find record from %s ...
Found %d record(s) for user %s
#2 - Cleaning Up....
You cannot delete %d records if only %d
exist.
Trouble cleaning up %s.
```

```

Cleaning up %d record(s) from %s ... OK
/var/run/utmp
/var/log/lastlog
Cleaning up %s ... OK
Cleaning up /var/log/messages ... OK
echo >/var/log/messages
Cleaning up /var/log/secure ... OK
echo >/var/log/secure
Cleaning up /var/log/maillog ... OK
echo >/var/log/maillog
Cleaning up /var/log/xferlog ... OK
echo >/var/log/xferlog
#3 - Successfull Cleaning up...
[root@agdrd19 bin]#

```

Next to be checked was "strings ". Running 'strings' on 'strings ' sounds interesting, and it did turn out interesting too! 'strings ' was in-fact the original string program, and the original was replaced by the new one! This must have been reflected in the change time stamps, which were not preserved properly. Since 'strings ' was the original, the next thing that came to my mind was too look for the modified program strings which must have been part of the rootkit.

```

[root@agdrd19 bin]# strings './strings'
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
strcpy
printf
fgets
__cxa_finalize
system
feof
malloc
__deregister_frame_info
setgid
strcmp
sprintf
fclose
fopen
_IO_stdin_used
__libc_start_main
strlen
setuid
__register_frame_info
GLIBC_2.1.3
GLIBC_2.1
GLIBC_2.0
PTRh`
QVh0
$Id: lsof v2.11 Exp $
[1;33m%s

```

```

/bin/su -
/usr/lib/libc/libah
egrep
%s|%s
%sps
%s %s
/usr/bin/strings' '
%s%s | /bin/egrep -v "%s"
[root@agdrd19 bin]#

```

From the output it seems as if the program is trying to hide (grep -v) a file "/usr/lib/libc/libah". This also points to the earlier output which detected a directory called /usr/lib/libc.

Examining the contents of /usr/lib/libc/ I found just one file called libph. It would have been interesting to see which program was accessing this file at the time of detection.

```

[root@agdrd19 libc]# ls -a
.  ..  libph
[root@agdrd19 libc]# cat libph
sendmail
eof
all
v
vadiml
sl2
sl3
slice
stream
stealth
startwu
superwu
7350wurm
x2
pscan
scan
r00t
php
[root@agdrd19 libc]# pwd
/root/analysis/images/testmounts/usr/lib/libc
[root@agdrd19 libc]#

```

OUTPUT FROM THE FINDS

All the output from the finds was stored in /root/search/

List of all the setuid/ setgid programs on the machine

```

[root@agdrd19 searches]# cat setuid-files
44231 65 -rwsr-xr-x 1 root root 65203 Mar 22 2001 ./sda10-dd/bin/mount
44232 34 -rwsr-xr-x 1 root root 33555 Mar 22 2001 ./sda10-dd/bin/umount
44247 24 -rwsr-xr-x 1 root root 22871 Jan 16 2001 ./sda10-dd/bin/su

```

```

12115 25 -r-sr-xr-x 1 root root 23719 Apr 7 2001 ./sda10-dd/sbin/pwdb_chkpwd
12116 25 -r-sr-xr-x 1 root root 24207 Apr 7 2001 ./sda10-dd/sbin/unix_chkpwd
12148 14 -rwxr-sr-x 1 root root 12919 Apr 7 2001 ./sda10-dd/sbin/netreport
32772 48 -rwsr-xr-x 1 root root 46523 Apr 4 2001 ./sda5-dd/bin/at
32828 44 -rwxr-sr-x 1 root kmem 44435 Feb 4 2001 ./sda5-dd/bin/man
32833 176 -rwxr-sr-x 1 root 14 176083 Feb 23 2001 ./sda5-dd/bin/minicom
32908 792 -rws--x--x 2 root root 803851 Mar 23 2001 ./sda5-dd/bin/suidperl
32908 792 -rws--x--x 2 root root 803851 Mar 23 2001 ./sda5-dd/bin/sperl5.6.0
32919 20 -rwxr-sr-x 1 root man 19883 Jan 6 2001 ./sda5-dd/bin/lockfile
32964 24 -rwsr-xr-x 1 root root 23091 Feb 5 2001 ./sda5-dd/bin/rcp
32966 20 -rwsr-xr-x 1 root root 19603 Feb 5 2001 ./sda5-dd/bin/rlogin
32967 20 -rwsr-xr-x 1 root root 16555 Feb 5 2001 ./sda5-dd/bin/rsh
33003 44 -rwsr-xr-x 1 root root 43347 Mar 9 2001 ./sda5-dd/bin/chage
33005 44 -rwsr-xr-x 1 root root 44987 Mar 9 2001 ./sda5-dd/bin/gpasswd
33017 36 -rwxr-sr-x 1 root fax 33267 Feb 26 2001 ./sda5-dd/bin/slocate
33141 24 -r-s--x--x 1 root root 22295 Jul 12 2000 ./sda5-dd/bin/passwd
33744 24 -rws----- 1 root root 21807 Apr 8 2001 ./sda5-dd/bin/chfn
33745 24 -rws--x--x 1 root root 21359 Apr 8 2001 ./sda5-dd/bin/chsh
33763 16 -rws----- 1 root root 14219 Apr 8 2001 ./sda5-dd/bin/newgrp
33774 20 -rwxr-sr-x 1 root tty 17451 Apr 8 2001 ./sda5-dd/bin/write
33805 204 -rwsr-xr-x 1 root root 204231 Apr 8 2001 ./sda5-dd/bin/ssh
33821 32 -rwsr-xr-x 1 root root 30071 Mar 8 2001 ./sda5-dd/bin/crontab
34122 16 -rwsr-xr-x 1 root root 16059 Apr 3 2001 ./sda5-dd/bin/kcheckpass
34131 68 -rwxr-sr-x 1 root root 64159 Apr 3 2001 ./sda5-dd/bin/kdesud
34289 40 -r-sr-x--- 1 root proxy 37971 Feb 14 2001 ./sda5-dd/bin/inndstart
34315 68 -r-sr-x--- 1 uucp proxy 62701 Feb 14 2001 ./sda5-dd/bin/rnews
34328 36 -r-sr-x--- 1 root proxy 34323 Feb 14 2001 ./sda5-dd/bin/startinfeed
35385 92 ---s--x--x 1 root root 89779 Feb 23 2001 ./sda5-dd/bin/sudo
37189 24 -rwsrwxrwx 1 root root 24073 Jul 26 05:07 ./sda5-dd/bin/rem
129699 20 -rws----- 1 root root 18256 Dec 1 2000 ./sda5-dd/sbin/traceroute
129700 8 -rwxr-sr-x 1 root voice 6584 Jul 13 2000 ./sda5-dd/sbin/utempter
133773 424 -r-sr-xr-x 1 root root 426587 Aug 28 2003 ./sda5-dd/sbin/sendmail
130034 12 -rwxr-sr-x 1 root voice 9180 Mar 16 2001 ./sda5-dd/sbin/gnome-pty-
        helper
130303 8 -rwsr-xr-x 1 root root 6392 Apr 7 2001 ./sda5-dd/sbin/usernetctl
130408 24 -rws--x--x 1 root root 20696 Feb 14 2001 ./sda5-dd/sbin/userhelper
132505 12 -r-s--x--- 1 root 48 10976 Mar 29 2001 ./sda5-dd/sbin/suexec
33878 8 -rws--x--x 1 root root 6040 Mar 30 2001 ./sda5-dd/X11R6/bin/Xwrapper
523304 20 ---x--s--x 1 501 500 17814 Oct 23 2003 ./sdb1-
        dd/sysadmin/Access_Logs/access-date.exe
295546 20 ---s--x--x 1 1054 300 17717 Aug 5 2003 ./sdb1-
        dd/spc/vimala/setids/access.exe
457862 4 drwxr-sr-x 2 root staff 4096 Mar 21 1999 ./sdb1-dd/ftp/pub
165141 8 -r-sr-xr-x 1 root root 5432 Mar 2 2001 ./sdb1-dd/gotcha/testme
131336 4 drwxr-sr-x 2 root root 4096 Mar 22 2001 ./sdc1-dd/ftp/pub
[root@agdr19 searches]# cat setuid-files | sed 's/ */ /g'
44231 65 -rwsr-xr-x 1 root root 65203 Mar 22 2001 ./sda10-dd/bin/mount
44232 34 -rwsr-xr-x 1 root root 33555 Mar 22 2001 ./sda10-dd/bin/umount
44247 24 -rwsr-xr-x 1 root root 22871 Jan 16 2001 ./sda10-dd/bin/su
12115 25 -r-sr-xr-x 1 root root 23719 Apr 7 2001 ./sda10-dd/sbin/pwdb_chkpwd

```

```

12116 25 -r-sr-xr-x 1 root root 24207 Apr 7 2001 ./sda10-dd/sbin/unix_chkpwd
12148 14 -rwxr-sr-x 1 root root 12919 Apr 7 2001 ./sda10-dd/sbin/netreport
32772 48 -rwsr-xr-x 1 root root 46523 Apr 4 2001 ./sda5-dd/bin/at
32828 44 -rwxr-sr-x 1 root kmem 44435 Feb 4 2001 ./sda5-dd/bin/man
32833 176 -rwxr-sr-x 1 root 14 176083 Feb 23 2001 ./sda5-dd/bin/minicom
32908 792 -rws--x--x 2 root root 803851 Mar 23 2001 ./sda5-dd/bin/suidperl
32908 792 -rws--x--x 2 root root 803851 Mar 23 2001 ./sda5-dd/bin/sperl5.6.0
32919 20 -rwxr-sr-x 1 root man 19883 Jan 6 2001 ./sda5-dd/bin/lockfile
32964 24 -rwsr-xr-x 1 root root 23091 Feb 5 2001 ./sda5-dd/bin/rcp
32966 20 -rwsr-xr-x 1 root root 19603 Feb 5 2001 ./sda5-dd/bin/rlogin
32967 20 -rwsr-xr-x 1 root root 16555 Feb 5 2001 ./sda5-dd/bin/rsh
33003 44 -rwsr-xr-x 1 root root 43347 Mar 9 2001 ./sda5-dd/bin/chage
33005 44 -rwsr-xr-x 1 root root 44987 Mar 9 2001 ./sda5-dd/bin/gpasswd
33017 36 -rwxr-sr-x 1 root fax 33267 Feb 26 2001 ./sda5-dd/bin/slocate
33141 24 -r-s--x--x 1 root root 22295 Jul 12 2000 ./sda5-dd/bin/passwd
33744 24 -rws----- 1 root root 21807 Apr 8 2001 ./sda5-dd/bin/chfn
33745 24 -rws--x--x 1 root root 21359 Apr 8 2001 ./sda5-dd/bin/chsh
33763 16 -rws----- 1 root root 14219 Apr 8 2001 ./sda5-dd/bin/newgrp
33774 20 -rwxr-sr-x 1 root tty 17451 Apr 8 2001 ./sda5-dd/bin/write
33805 204 -rwsr-xr-x 1 root root 204231 Apr 8 2001 ./sda5-dd/bin/ssh
33821 32 -rwsr-xr-x 1 root root 30071 Mar 8 2001 ./sda5-dd/bin/crontab
34122 16 -rwsr-xr-x 1 root root 16059 Apr 3 2001 ./sda5-dd/bin/kcheckpass
34131 68 -rwxr-sr-x 1 root root 64159 Apr 3 2001 ./sda5-dd/bin/kdesud
34289 40 -r-sr-x--- 1 root proxy 37971 Feb 14 2001 ./sda5-dd/bin/inndstart
34315 68 -r-sr-x--- 1 uucp proxy 62701 Feb 14 2001 ./sda5-dd/bin/rnews
34328 36 -r-sr-x--- 1 root proxy 34323 Feb 14 2001 ./sda5-dd/bin/startinnfeed
35385 92 ---s--x--x 1 root root 89779 Feb 23 2001 ./sda5-dd/bin/sudo
37189 24 -rwsrwxrwx 1 root root 24073 Jul 26 05:07 ./sda5-dd/bin/rem
129699 20 -rws----- 1 root root 18256 Dec 1 2000 ./sda5-dd/sbin/traceroute
129700 8 -rwxr-sr-x 1 root voice 6584 Jul 13 2000 ./sda5-dd/sbin/utempter
133773 424 -r-sr-xr-x 1 root root 426587 Aug 28 2003 ./sda5-dd/sbin/sendmail
130034 12 -rwxr-sr-x 1 root voice 9180 Mar 16 2001 ./sda5-dd/sbin/gnome-pty-
        helper
130303 8 -rwsr-xr-x 1 root root 6392 Apr 7 2001 ./sda5-dd/sbin/usernetctl
130408 24 -rws--x--x 1 root root 20696 Feb 14 2001 ./sda5-dd/sbin/userhelper
132505 12 -r-s--x--- 1 root 48 10976 Mar 29 2001 ./sda5-dd/sbin/suexec
 33878 8 -rws--x--x 1 root root 6040 Mar 30 2001 ./sda5-dd/X11R6/bin/Xwrapper
523304 20 ---x--s--x 1 501 500 17814 Oct 23 2003 ./sdb1-
        dd/sysadmin/Access_Logs/access-date.exe
295546 20 ---s--x--x 1 1054 300 17717 Aug 5 2003 ./sdb1-
        dd/spc/vimala/setids/access.exe
457862 4 drwxr-sr-x 2 root staff 4096 Mar 21 1999 ./sdb1-dd/ftp/pub
165141 8 -r-sr-xr-x 1 root root 5432 Mar 2 2001 ./sdb1-dd/gotcha/testme
131336 4 drwxr-sr-x 2 root root 4096 Mar 22 2001 ./sdc1-dd/ftp/pub
[root@agdrd19 searches]#
None of the files are seen modified in 2004.
[root@agdrd19 searches]# cat dot-dot-dot
-rw-r--r-- 1 500 300 1584 1999-12-17 00:50 ./sdb1-
dd/spc/patil/.enlightenment/...e_session-XXXXXX

```



```

-rw-r--r--  1 500   300      1142 1999-12-17 00:50 ./sdb1-
dd/spc/patil/.enlightenment/...e_session-XXXXXX.clients.0
-rw-r--r--  1 500   300      0 1999-12-17 00:50 ./sdb1-
dd/spc/patil/.enlightenment/...e_session-XXXXXX.snapshots.0
-rw-r--r--  1 501   300     1584 2000-07-07 11:59 ./sdb1-
dd/spc/rk/.enlightenment/...e_session-XXXXXX
-rw-r--r--  1 501   300      0 2000-07-07 11:59 ./sdb1-
dd/spc/rk/.enlightenment/...e_session-XXXXXX.snapshots.0
-rw-r--r--  1 501   300     1142 2000-07-07 11:59 ./sdb1-
dd/spc/rk/.enlightenment/...e_session-XXXXXX.clients.0
-rw-----  1 509   300      715 2004-04-05 10:44 ./sdb1-dd/spc/srinivas/...
-rw-r--r--  1 1003  321      362 2003-04-24 03:25 ./sdb1-
dd/shakuni/shree/doc/metadata/...txt
-rw-r--r--  1 1052  321      105 2003-09-12 03:43 ./sdb1-
dd/shakuni/shikha/proj/...proc
-rw-r--r--  1 1050  505     359 2003-04-24 05:45 ./sdb1-
dd/pt/rpredy/public-html/metadata/...txt
drwxr-xr-x  5 50103  321     4096 2004-08-01 07:17 ./sdb1-
dd/diva/.kde/tmp/var/...
[root@agdrd19 searches]#

```

The output shows that there are a few files with the name “...” in some of the users directories. But there is one suspicious file(a directory) residing in the home directory of the user diva. This was noted, for further analysis

Next was the file superwu

```

[root@agdrd19 searches]# cat superwu
./sda10-dd/lib/security/www/superwu
[root@agdrd19 searches]#

```

There seems to be a file named superwu in a very peculiar location: /lib!! (sda10 was /). I immediately used the file command to find what it was:

```

[root@agdr19 testmounts]# file ./sda10-dd/lib/security/www/superwu
./sda10-dd/lib/security/www/superwu: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), statically linked, corrupted section header size
[root@agdr19 testmounts]#

```

The output of the file command shows that it was an executable. Being a 32-bit on Intel 80386 gives a hint it must have been compiled on a Linux machine.

Going ahead with the search for user files, starting with the user ro:

```

[root@agdrd19 searches]# cat uid-ro
./sdb1-dd/ro
./sdb1-dd/ro/.bash_logout
./sdb1-dd/ro/.bash_profile
./sdb1-dd/ro/.bashrc
./sdb1-dd/ro/Desktop
./sdb1-dd/ro/Desktop/kontrol-panel
./sdb1-dd/ro/Desktop/.directory
./sdb1-dd/ro/Desktop/Linux Documentation
./sdb1-dd/ro/Desktop/www.redhat.com
./sdb1-dd/ro/Desktop/Printer
./sdb1-dd/ro/.kde

```

```
./sdb1-dd/ro/.kde/Autostart
./sdb1-dd/ro/.kde/Autostart/.directory
./sdb1-dd/ro/.emacs
./sdb1-dd/ro/.screenrc
[root@agdrd19 searches]#
```

The output does not show any un-nerving fact, and it looks more like a default content of a home directory of a newly added user. This output also helps to guarantee the fact that the user ro was added only few hours before being detected.

Next was the user ravi:

```
[root@agdr19 searches]# cat uid-ravi
./sdb1-dd/ravi
./sdb1-dd/ravi/.bash_logout
./sdb1-dd/ravi/.bash_profile
./sdb1-dd/ravi/.bashrc
./sdb1-dd/ravi/Desktop
./sdb1-dd/ravi/Desktop/kontrol-panel
./sdb1-dd/ravi/Desktop/.directory
./sdb1-dd/ravi/Desktop/Linux Documentation
./sdb1-dd/ravi/Desktop/www.redhat.com
./sdb1-dd/ravi/Desktop/Printer
./sdb1-dd/ravi/.kde
./sdb1-dd/ravi/.kde/Autostart
./sdb1-dd/ravi/.kde/Autostart/.directory
./sdb1-dd/ravi/.kde/.var
./sdb1-dd/ravi/.kde/.var/ps
./sdb1-dd/ravi/.kde/.var/ps/lang
./sdb1-dd/ravi/.kde/.var/ps/lang/english.lng
./sdb1-dd/ravi/.kde/.var/ps/daemon
./sdb1-dd/ravi/.kde/.var/ps/ps
./sdb1-dd/ravi/.kde/.var/ps/mkconf
./sdb1-dd/ravi/.kde/.var/ps/log
./sdb1-dd/ravi/.kde/.var/ps/log/ps.log
./sdb1-dd/ravi/.kde/.var/ps/log/USER1.TRL
<SOME MORE FILES IN LOG>
./sdb1-dd/ravi/.kde/.var/ps/help/ADDLOG.TXT
./sdb1-dd/ravi/.kde/.var/ps/help/DELLOG.TXT
./sdb1-dd/ravi/.kde/.var/ps/help/LISTLOGS.TXT
./sdb1-dd/ravi/.kde/.var/ps/help/DCCSENDME.DEU
<LOT MORE FILES IN HELP>
./sdb1-dd/ravi/.kde/.var/ps/ps.pid
./sdb1-dd/ravi/.kde/.var/ps/daemon.old
./sdb1-dd/ravi/.emacs
./sdb1-dd/ravi/.screenrc
./sdb1-dd/ravi/.ispoof
./sdb1-dd/ravi/.oidentd.conf
./sdb1-dd/ravi/.bash_history
./sdc1-dd/spool/mail/ravi
```

```
[root@agdr19 searches]#
```

Apart from showing a bad and hidden directory name(./sdb1-dd/ravi/.kde/.var), it showed that the user ravi had downloaded a utility named ps. His idea must have been to overwrite the ps that is already present on the system. The ps.pid must have been to store the id of some daemon that the ps would hide. Also there was another file called /.oidentd.conf which sounds like some configuration file of some daemon. A web search revealed that this was a configuration file for *oidentd*¹⁰: As per the website, oidentd is a comfortable identd daemon, which makes it possible to spoof identd requests. It is supported by many bouncers like psybnc. Spoof means that it is for each user possible to send his own ident back.

This is above all an advantage for the quakenet, because there is a trust limited to 4 idents.

Analysing for the user diva, the following results were seen:

```
[root@agdr19 searches]# cat uid-diva
```

```
./sdb1-dd/diva
./sdb1-dd/diva/.bash_logout
./sdb1-dd/diva/.bash_profile
./sdb1-dd/diva/.bashrc
./sdb1-dd/diva/Desktop
./sdb1-dd/diva/Desktop/kontrol-panel
./sdb1-dd/diva/Desktop/.directory
./sdb1-dd/diva/Desktop/Linux Documentation
./sdb1-dd/diva/Desktop/www.redhat.com
./sdb1-dd/diva/Desktop/Printer
./sdb1-dd/diva/.kde
./sdb1-dd/diva/.kde/Autostart
./sdb1-dd/diva/.kde/Autostart/.directory
./sdb1-dd/diva/.kde/tmp
./sdb1-dd/diva/.kde/tmp/var
./sdb1-dd/diva/.kde/tmp/var/...
./sdb1-dd/diva/.kde/tmp/var/.../Unreal3.1.3
./sdb1-dd/diva/.kde/tmp/var/.../Unreal3.1.3/CVS
```

<LOT MORE FILES INDICATING INSTALLATION OF THE COMPLETE SOFTWARE!>

```
./sdb1-dd/diva/.kde/tmp/var/.../services
./sdb1-dd/diva/.kde/tmp/var/.../services/services
./sdb1-dd/diva/.kde/tmp/var/.../services/listnicks
./sdb1-dd/diva/.kde/tmp/var/.../services/listchans
./sdb1-dd/diva/.kde/tmp/var/.../services/languages
./sdb1-dd/diva/.kde/tmp/var/.../services/languages/cat
./sdb1-dd/diva/.kde/tmp/var/.../services/exception.db
./sdb1-dd/diva/.kde/tmp/var/.../.sh
./sdb1-dd/diva/.kde/tmp/var/.../epona-1.4.14
./sdb1-dd/diva/.kde/tmp/var/.../epona-1.4.14/BUGS
```

<MANY MORE FILES, AGAIN A COMPLETE SOFTWARE INSTALLATION>

```
./sdb1-dd/diva/.emacs
```

¹⁰ <http://psybnc.cynapses.org/?lang=en&show=oidentd>

```
./sdb1-dd/diva/.screenrc
./sdb1-dd/diva/.bash_history
[root@agdr19 searches]#
```

Again the user diva contained two software, epona-1.4.14 and Unreal3.1.3 installed in the directory ./sdb1-dd/diva/.kde/tmp/var/.../. Both the softwares are related to IRC, Internet Relay Chat. Epona¹¹ is a set of services for IRC networks that allows users to manage their nicks and channels in a secure and efficient way, and administrators to manage their network with powerful tools. Unreal3.1.3 is a an IRC Daemon named UnrealIRCd¹².

The next approach was look for all the logs of the system. To know whats happening in the security logs, I did a grep -v 172.16 to avoid all the logs that were from the internal IP address. On 5th of August there were attempts being made to log into the server using ssh. The user tried was **test/guest**. The remote ip, in such a case plays a vital role in identifying the attacker. The following logs obtained from /var/log/secure.2 file show the following output

```
Directory: /var//log/
Image: images/sdc1-dd Meta: 98497
```

```
r / r      secure.2
 2004.08.05 23:24:36 (IST)
 2004.08.09 13:37:24 (IST)
2004.10.26 11:01:21 (IST)
27011      0      0      98590y
```

The above file secure.2 contained following important ips:

```
Aug  4 08:49:36 draupadi xinetd[679]: START: telnet pid=17700
from=193.109.122.10
```

```
Aug  5 19:47:30 draupadi sshd[6394]: input_userauth_request: illegal user test
Aug  5 19:47:30 draupadi sshd[6394]: Failed password for illegal user test from
210.201.0.61 port 54499 ssh2
Aug  5 19:47:31 draupadi sshd[6394]: Received disconnect from 210.201.0.61:
11: Bye Bye
Aug  5 19:47:34 draupadi sshd[6411]: input_userauth_request: illegal user
guest
Aug  5 19:47:34 draupadi sshd[6411]: Failed password for illegal user guest
from 210.201.0.61 port 54509 ssh2
Aug  5 19:47:34 draupadi sshd[6411]: Received disconnect from 210.201.0.61:
11: Bye Bye
```

```
Aug  5 20:19:00 draupadi xinetd[679]: START: telnet pid=11261
from=193.109.122.13
```

```
Aug  5 21:58:27 draupadi sshd[19651]: Did not receive identification string from
81.18.78.97.
```

I searched these IP addressses on the web. And found them to belonging to a

¹¹ <http://www.epona.org/>

¹² <http://www.unrealircd.com/>

location in Netherlands APNIC¹³ didn't have Information about the IP's seen in secure logs But RIPE¹⁴ did!

It was an IP from Netherlands, New Zealand

inetnum: 193.109.122.0 - 193.109.122.255
netname: BIT-IRC-1
descr: BIT proxyscan PI space
country: NL
route: 193.109.122.0/24
descr: route object for 193.109.122.0/24
origin: AS12859
notify: scancomplaints@bit.nl
mnt-by: BIT-MNT
changed: scancomplaints@bit.nl 20020122
source: RIPE

person: Sabri Berisha
address: Postbus 536
address: 6710 BM EDE
address: The Netherlands
phone: +31 318 695917
e-mail: scancomplaints@bit.nl
nic-hdl: SB825-RIPE
mnt-by: BIT-MNT
changed: scancomplaints@bit.nl 20020122
source: RIPE

Searching for the user ravi, I came across a peculiar file in his account directory.

File: /home/ravi/.kde/.var/.sh

Image: images/sdb1-dd Meta: 441687

This file contains the log for the irc daemon the attacker was running!

Referring to the earlier search result that showed me a directory named "...", I checked up the same file and its presence on the timeline. The timeline showed that this directory was being used by the user diva for running an IRC Server

superwu

This was one thing that could never be neglected. Arjuna had noticed a process named superwu running on the system. It was a command top(1) that showed him the process. Unfortunately no other information was available on the run time capabilities of the process. Earlier find had showed the program being in

¹³ APNIC is one of four Regional Internet Registries currently operating in the world. It provides allocation and registration services which support the operation of the Internet globally. It is a not-for-profit, membership-based organisation whose members include Internet Service Providers, National Internet Registries, and similar organisations. APNIC represents the Asia Pacific region, comprising 62 economies. <http://www.apnic.net/>

¹⁴ The RIPE NCC is an independent, not-for-profit membership organisation that supports the infrastructure of the Internet through technical co-ordination in its service region. The most prominent activity of the RIPE NCC is to act as the [Regional Internet Registry \(RIR\)](#) providing global Internet resources and related services ([IPv4, IPv6 and AS Number resources](#)) to members in the RIPE NCC service region. <http://www.ripe.net/>

the directory, “/root/analysis/images/testmounts/sda10-dd/lib/security/www”. The directory name itself is peculiar, but easily ignored by a casual viewer.

A study of this directory revealed a bouncer. A bouncer is a program that works similar to a proxy. It accepts connections from one location and *forwards* the connection to another location. A bouncer can be used to *hide* one’s true IP address. One can connect to a bouncer and ask it to forward connection to another destination. The destination will in turn get the source address as that of the bouncer. Bouncers can be used to perform attacks on the system by hiding one’s location. The attacked machine will see the attack being happening from the address of the bouncer.

The directory contained a bouncer named, psybnc¹⁵. The bouncer software was found in a hidden directory named “.bash”. The attacker got this software by means of a gzipped¹⁶ tar¹⁷ archive.

```
[root@agdr19 www]# tar -ztf bnc.tgz
```

```
.bash/  
.bash/key/  
.bash/key/psybnc.cert.pem  
.bash/key/psybnc.key.pem  
.bash/key/psybnc.req.pem  
.bash/log/  
.bash/log/INFO  
.bash/log/USER1.LOG  
.bash/log/USER1.TRL  
.bash/log/psybnc.log
```

<some more files related to psybnc bouncer software>

```
[root@agdr19 www]#
```

A simple search for the two keywords “sniff” and “promisc” showed presence of two executables that would attempt to sniff packets from the network. The output is discussed in the section, String Search later in this paper.

This directory also contained a file named, firewall. The find(1) command revealed this to be a bash script. Following is the output of the command file and the contents of the script.

```
[root@agdr19 www]# file firewall
```

```
firewall: Bourne shell script text executable
```

```
[root@agdr19 www]# cat firewall
```

```
#!/bin/sh
```

```
#!/bin/bash
```

```
#
```

```
# This File Will Block All Existent Ports From 15000 To 65536
```

```
# Don't Forget To Modificate My Port If You Will Use This FireWall
```

```
# Be Carefully Use This File Only On Your Risk
```

```
# File Created By NaRciS
```

¹⁵ <http://www.psychoid.net/psybnc.html>

¹⁶ gzip is a gnu[10] implementation of a file compressor that reduces files by using Lempel-Ziv coding (LZ77)

¹⁷ tar is an archiving program designed to store and extract files from an archive file known as a tarfile.

```

#

BLK=""
RED=""
GRN=""
YEL=""
BLU=""
MAG=""
CYN=""
WHI=""
DRED=""
DGRN=""
DYEL=""
DBLU=""
DMAG=""
DCYN=""
DWHI=""
RES=""
FIREWALLLOG=/lib/security/www/firewall.log

if [ -d /lib/security/www/ ];then

echo "${DGRN}[-] ${CYN}/lib/security/www/.rd/ ${GRN}Exist Not Creating
${CYN}...      ${DGRN}[-]${RES}"
echo
else
echo "${DGRN}[-] ${CYN}/lib/security/www/.rd/ ${RED}Does Not Exist Creating
${CYN}...      ${DGRN}[-]${RES}"
echo
mkdir -p /lib/security/www/

fi

if [ -f $FIREWALLLOG ];then
rm -rf $FIREWALLLOG
touch $FIREWALLLOG
else
touch $FIREWALLLOG

fi

if [ -f /sbin/ipchains ];then

echo "/sbin/ipchains STATUS : [OK]" >> $FIREWALLLOG
echo "Tring To Start And Aply FireWall Rules For NaRciS Settings" >>
$FIREWALLLOG
echo "${DGRN}[-] ${CYN}Well I Have A Good New : /sbin/ipchains
${GRN}EXISTSS${CYN} ...      ${DGRN}[-]${RES}"
echo
echo "${DGRN}[-] ${CYN}Tring To Start And Aply FireWall Rules For

```

```

${YEL}NaRciS ${CYN}Settings ... ${DGRN}[-]${RES}"
echo "Displaing Status Of /sbin/ipchains -F Before Installing The FireWall" >>
$FIREWALLLOG
echo
"#####" >> $FIREWALLLOG
echo "#####
Status#####" >> $FIREWALLLOG
/sbin/ipchains -L -n >> $FIREWALLLOG
echo "##### END Of The Status Before Installing The FireWall
#####" >> $FIREWALLLOG
/sbin/ipchains -F >> $FIREWALLLOG
/sbin/ipchains -I input -j DENY -s 0/0 -d 0/0 -p tcp --destination-port 15000:65535
>> $FIREWALLLOG
/sbin/ipchains -I input -j ACCEPT -s 0/0 -d 0/0 -p tcp --sport 216 >>
$FIREWALLLOG
/sbin/ipchains -I input -j ACCEPT -s 0/0 -d 0/0 -p tcp --dport 216 >>
$FIREWALLLOG
/sbin/ipchains -I input -j ACCEPT -s 0/0 -d 0/0 -p udp --sport 216 >>
$FIREWALLLOG
/sbin/ipchains -I input -j ACCEPT -s 0/0 -d 0/0 -p udp --dport 216 >>
$FIREWALLLOG
/sbin/ipchains -I output -j ACCEPT -s 0/0 -d 0/0 -p tcp --sport 216 >>
$FIREWALLLOG
/sbin/ipchains -I output -j ACCEPT -s 0/0 -d 0/0 -p tcp --dport 216 >>
$FIREWALLLOG
/sbin/ipchains -I output -j ACCEPT -s 0/0 -d 0/0 -p udp --sport 216 >>
$FIREWALLLOG
/sbin/ipchains -I output -j ACCEPT -s 0/0 -d 0/0 -p udp --dport 216 >>
$FIREWALLLOG
echo "Displaing Status Of /sbin/ipchains -F After Installing The FireWall " >>
$FIREWALLLOG
echo "#####
Status#####" >> $FIREWALLLOG
/sbin/ipchains -L >> $FIREWALLLOG
echo "##### END Of The Status After Installing The FireWall #####"
>> $FIREWALLLOG
echo
"#####" >> $FIREWALLLOG
else

echo "/sbin/ipchains STATUS : [FAILED]" >> $FIREWALLLOG
echo "${DGRN}[-] ${CYN}Well I Have A Bad New : /sbin/ipchains
${RED}MISSING${CYN} ...      ${DGRN}[-]${RES}"

fi

#
# EOF

```


#

[root@agdr19 www]#

The script develops a firewall that blocks the ports from 15000 to 65535

There was another tar archive name windmilk.tgz that contained the executable superwu. It was not possible to determine what the executable superwu did.

© SANS Institute 2000 - 2005, Author retains full rights.

Timeline Analysis

Note: Detailed timeline is available separately along with this submission.

I prefer the Autopsy forensic browser for generating timelines. That's mainly because of the ease of operation and managing the complete study as such. Moreover the browser (or server as it should rightly be referred as) also has facility to make notes which come in very handy during the reporting phase of the analysis. In my report, I have put in excerpts from the notes that I made on Autopsy.

I generated the time line for the month of July and August. It shows a steady activity of the suspicious user ravi from the 26th of July itself.

```
Mon Jul 26 2004 14:52:16 17 m.c l/lrwxrwxrwx root root 441677
                        /home/ravi/Desktop/Autostart ->
                        ../.kde/Autostart
```

The above timeline record shows the date when this account got created.

On the 30th of July the timeline shows the account of user diva being created. Again this is the first instance of the user CREATING a file. Modification time stamps can get transferred when the file is transferred, but the creation time stamp shows presence on the machine.

```
Fri Jul 30 2004 12:09:56 4096 m.. d/drwxr-xr-x diva shakuni 49094
                        /home/diva/Desktop
                        224 m.. -/-rw-r--r-- diva shakuni 217
                        /home/diva/.bash_profile
                        306 m.. -/-rw-r--r-- diva shakuni 49262
                        /home/diva/Desktop/.directory
                        107 m.. -/-rw-r--r-- diva shakuni 49293
                        /home/shakuni/golani/1.jpg (deleted-realloc)
                        280 m.. -/-rw-r--r-- diva shakuni 49297
                        /home/shakuni/srinivas/public-html/temp/temp
                        (deleted-realloc)
                        280 m.. -/-rw-r--r-- diva shakuni 49297
                        /home/diva/Desktop/Printer
                        4096 m.. d/drwxr-xr-x diva shakuni 221
                        /home/diva/.kde/Autostart
                        4096 m.. -/drwxr-xr-x diva shakuni 49094
                        /home/shakuni/selva/mail/sudha.lock.108452841
                        1.22958.draupadi.ncb.ernet.in (deleted-
                        realloc)
                        4096 m.. -/drwxr-xr-x diva shakuni 49094
                        /home/shakuni/selva/mail/
                        support.lock.1084528416.22958.draupadi.ncb.er
                        net.in (deleted-realloc)
                        24 m.. -/-rw-r--r-- diva shakuni 216
                        /home/diva/.bash_logout
                        3728 m.. -/-rw-r--r-- diva shakuni 226
                        /home/diva/.screenrc
                        4096 m.. -/drwxr-xr-x diva shakuni 49094
                        /home/shakuni/selva/mail/d0253039 (deleted-
                        realloc)
                        80 m.. -/-rw-r--r-- diva shakuni 49290
                        /home/diva/Desktop/Linux Documentation
                        747 m.. -/-rw-r--r-- diva shakuni 224
                        /home/diva/.emacs
                        17 m.c l/lrwxrwxrwx root root 49289
                        /home/diva/Desktop/Autostart ->
                        ../.kde/Autostar
```

The time line also shows that the user diva compiled a code on the Server.

```
Fri Jul 30 2004 12:29:10 965749 m.. -/-rwx--x--x diva      sedb
16743
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/i
rcd (deleted-realloc)
39848 m.. -/-rw-r--r-- diva      sedb
16742
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/z
ip.o
46496 m.. -/-rw-r--r-- diva      sedb
16741
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/w
howas.o
965749 m.. -/-rwx--x--x diva      sedb
16743
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/s
ftp-server
42832 m.. -/-rw-r--r-- diva      sedb
16745
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/c
hkmatch.o
49880 m.. -/-rw-r--r-- diva      sedb
16744
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/c
hkcrule.o
Fri Jul 30 2004 12:29:12 0 mac -/-rw----- diva      sedb
92 /tmp/tmpf5A22n6 (deleted)
82477 m.. -/-rwxr-xr-x diva      sedb
16746
/home/diva/.kde/tmp/var/.../Unreal3.1.3/usr/c
hkconf
0 mac -rw----- diva      sedb 92
<sda9-dd-dead-92>
0 mac -/-rw----- diva      sedb
92 /tmp/ccFOMPGc.ld (deleted)
Fri Jul 30 2004 13:13:17 2900 m.. -/-rwx----- diva      shakuni
49355
/home/diva/.kde/tmp/var/.../Unreal3.1.3/ircd
```

The highlighted portion shows that there were some temporary files created to perform linking. The files were immediately deleted after the linking was done. The module being compiled was called **Unreal3.1.3** which is an irc(Internet Relay Chat) daemon.

Again on August 1st the attacker compiled another software, epona, version 1.4.14. As per the website, Epona is a set of services for IRC networks that allows users to manage their nicks and channels in a secure and efficient way, and administrators to manage their network with powerful tools.

```
Sun Aug 01 2004 16:49:48 71484 m.. -/-rw-r--r-- diva      sedb
458539 /home/diva/.kde/tmp/var/.../epona-
1.4.14/messages.o
Sun Aug 01 2004 16:49:49 59560 m.. -/-rw-r--r-- diva      sedb
458540 /home/diva/.kde/tmp/var/.../epona-
1.4.14/misc.o
Sun Aug 01 2004 16:49:51 64704 m.. -/-rw-r--r-- diva      sedb
458541 /home/diva/.kde/tmp/var/.../epona-
1.4.14/news.o
Sun Aug 01 2004 16:49:58 142252 m.. -/-rw-r--r-- diva      sedb
458542 /home/diva/.kde/tmp/var/.../epona-
```

```

1.4.14/nickserv.o
Sun Aug 01 2004 16:50:05 54344 m.. -/-rw-r--r-- diva sedb
458544 /home/diva/.kde/tmp/var/.../epona-
1.4.14/process.o
138752 m.. -/-rw-r--r-- diva sedb
458543 /home/diva/.kde/tmp/var/.../epona-
1.4.14/operserv.o
Sun Aug 01 2004 16:50:15 720378 m.. -/-rwxr-xr-x diva shakuni
458561 /home/diva/.kde/tmp/var/.../epona-
1.4.14/services

```

Finally the compilation created an executable file named, services.

Based on the earlier media analysis that revealed a bounce software named, psybnc, the timeline revealed that the software was used on 5th of August.

```

Thu Aug 05 2004 20:16:05 206 m.. -/-rw----- root root 22156
/lib/security/www/.bash/log/psybnc.log.old
Thu Aug 05 2004 20:18:56 1578 m.. -/-rw----- root root 22172
/lib/security/www/.bash/motd/USER1.MOTD.old
Thu Aug 05 2004 20:19:17 669 m.. -/-rw----- root root 22158
/lib/security/www/.bash/psybnc.conf.old
Thu Aug 05 2004 20:19:19 669 m.. -/-rw----- root root 22165
/lib/security/www/.bash/psybnc.conf
Thu Aug 05 2004 20:19:32 1525 m.. -/-rw----- root root 22155
/lib/security/www/.bash/log/psybnc.log
Thu Aug 05 2004 20:19:34 1540 m.. -/-rw----- root root 22170
/lib/security/www/.bash/motd/USER1.MOTD

```

String Search

sniff

A search for the keywords, sniff and promisc in the directory “lib/security/www” of the compromised machine revealed two files. The output of the grep search is as follows:

```

[root@agdr19 www]# egrep -ira 'sniff|promisc' * | strings
read:# Sorts the output from LinSniffer 0.03 [BETA] by Mike Edulla
<medulla@infosoc.com>
read:          # The line is one of linsniffs "separator" lines
write:j
lh8
t(hv
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
[root@agdr19 www]# pwd
/root/analysis/images/testmounts/slash/lib/security/www
[root@agdr19 www]#

```

It showed two files, read and write that contained these keywords.

The first file, read is a perl script that uses LinSniffer that is a line sniffer. The second line has an error message saying “cant set promiscuous mode”. This indicates that the attacker did try to deploy the a packet sniffing code in draupadi!

Viewing the timestamps of these files reveal that the files were last modified on the 5th of August. The following output also shows the file, write being last

modified on the 26th of July. It could be possible that these executables could have been running on the system at the time of detection. The names, read and write could easily go unnoticed to an in-experienced eye.

```
[root@agdr19 www]# stat read
```

```
File: `read'
Size: 4060      Blocks: 8      IO Block: 4096  regular file
Device: 700h/1792d Inode: 42245  Links: 1
Access: (0755/-rwxr-xr-x) Uid: ( 0/  root) Gid: ( 0/  root)
Access: 2004-08-08 02:10:05.000000000 -0400
Modify: 2004-08-05 09:50:52.000000000 -0400
Change: 2004-08-07 19:46:22.000000000 -0400
```

```
[root@agdr19 www]# stat write
```

```
File: `write'
Size: 17960     Blocks: 38     IO Block: 4096  regular file
Device: 700h/1792d Inode: 42246  Links: 1
Access: (0755/-rwxr-xr-x) Uid: ( 0/  root) Gid: ( 0/  root)
Access: 2004-10-26 01:19:20.000000000 -0400
Modify: 2004-08-05 09:50:52.000000000 -0400
Change: 2004-08-07 19:46:22.000000000 -0400
```

```
[root@agdr19 www]#
```

./susu

A look at the .bash_history of the user ravi revealed that the user was on the system for a long time. So there were chances that I could find lot of traces of activities being done by the user. In the history file, there was a line “./susu”. I did a keyword search for this and found that this was not an executable.

Search for "susu"

Fragment: 640956

Allocated

Group: 19

Hide Meta Data Address

Pointed to by Inode: 49303

Pointed to by file: /home/diva/.kde/tmp/var/.../Unreal3.1.3/ircd.conf

Fragment 640956 (Hex - Ascii)

67: 709 (X:susu1:sus)

68: 715 (usu1:susu2)/home/diva/.kde/tmp/var/.../Unreal3.1.3/ircd.conf

Contents found:

X:LINE Die/Restart Password

X:susu1:susu2

#####

Again this was found in the IRCd daemon logs. The X in the beginning of the configuration file seems to indicate it as an encrypted password. All other searches for the word susu resulted in some character combinations, that

looked more like *MIME BASE64*¹⁸ encoding.

Sniffer

A sniffer was found while searching for one. However it was found in one of the user directories. On asking Mr. Arjuna he confirmed that it was a legitimate code, and the user too agreed that he had downloaded the code.

Superwu

Detailed analysis on this is available in Media Analysis section.

Conclusions

Examination of the system revealed modified binaries and ircd daemon deployed by the attacker. The sequence of activities show that the system was *available* to the attacker for a long duration of two weeks before being detected.

The attacker seems to be a experienced one and knows where and how to hide files. The locations `/usr/lib/libc`, `/usr/bin/strings` “ show the attacker’s knowledge about the system.

The username created for the first time with proper Name shows that she had his presence of mind in place and was cautious before doing nasty things inside the server. As she spent time on Draupadi, she realized that her activities are going un-noticed. That made her careless and made her commit serious blunders enough to get detected by a simple regular user.

Motive

The presences of two IRC daemons indicate that she did have some more interests in the machine other than just a compromise. The intention was to run IRC daemons. There are instance where these daemons were started too.

The attacker did not have intentions to gain information from the server. No attempts were found to destroy content or search information.

But the presence of a log-eraser `/usr/bin/rem`” indicate that she was ready to erase all logs in case of a tight situation. Being gone undetected for a duration of nearly two weeks, the attacker was careless enough of not coming to know that she was spotted. Had she noticed that simple programs like `ls` are crashing, she should have cleared all the logs.

What caused the compromise?

Its not clear, how the attacker gained access to Draupadi, not yet. Mr. Arjuna has a feeling that he might have made a mistake while updating the router ACLs and might have left the server open in the Internet. This would open all the services Draupadi was running to be accessible from the Internet, and would make the server potentially vulnerable to exploits.

¹⁸ MIME (Multipurpose Internet Mail Extensions). The Base64 encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. A 65-character subset ([A-Za-z0-9+/=]) of US-ASCII is used, enabling 6 bits to be represented per printable character. As per MIME::Base64(3pm).

References

- [1] The link that helped me find out the IP address.
http://www.ripe.net/perl/whois?form_type=simple&full_query_string=&searchtext=193.109.122.10&do_search=Search
- [2] The website that gave information about psybnc.
<http://www.psychoid.net/psybnc.html>
- [3] Information about oidentd
<http://psybnc.cynapses.org/?lang=en&show=oidentd>.
- [4] The official website for UnRealIRCd, an IRC Daemon
<http://www.unrealircd.com/>
- [5] All information about epona an extension for IRC daemons
<http://www.epona.org/>
- [6] Asia Pacific Network Information Centre <http://www.apnic.net/>
- [7] The Whois search page that identified the IP Address attempting to make telnet connection to draupadi <http://www.ripe.net/perl/whois>
- [8] Official website for TSK and Autopsy: <http://www.sleuthkit.org/>
- [9] Some very informative and enlightening forensic analysis played important role in gathering knowledge for this paper. www.honeynet.org
- [10] GNU is Not UNIX. A compilation of all the binaries that are found on the GNU/Linux machine www.gnu.org
- [11] All web searches: www.google.com

APPENDIX A

EXPLANATIONS OF UTILITIES AND COMMANDS

File-System Basics

A file-system is nothing but a data-structure that defines how data(bits and bytes) can be stored on the hard disks. To let the user use this data structures, a file-system defines set of functionalities(APIs) that can modify the data structures and the data, in a consistent and a efficient manner.

Its the file-system that determines how much amount of space is available free on the computer, and where is it located. Its the file-system again that decides where and how much amount of space to be allocated for a file of size 23 bytes and for a 56Mb file. To address different requirements and different working conditions, there were various file-systems developed over a period of time.

eg.

Consider a Linux server that is being shared by some 10-20 team members of a department. In such a case, everyone would prefer to store data separately from one another, grant/restrict access to others. This demands a file-system that can understand users, and let them control permissions. In that case, we could use a file-system called ext2 or ext3.

However when the time comes to transfer data on a floppy one cannot assure that the same user related information will be available at the machine where the floppy is being taken. This would ask for a file-system that permits access to everyone! So we have file-systems like VFAT, FAT32 etc.

To let the users organize their files, nearly all file-systems provide a concept called directory. A directory can be viewed as a collection of files. It contains nothing but list of files that it contains! Moreover directories can contain sub-directories too.

To keep a track of which area of the disk is occupied, and by which file, the file-system maintains its data-structure in a form of inodes and blocks/clusters. Inodes store information for each file and the address where the data for that file is stored, where as the cluster information stores how much, what part of area is free etc. The minimal amount of disk space one can access is called block.

When a file is deleted, the inode that it was using, is simply marked un-used, and the data it was holding is marked as free! Note that no attempts is made to *erase* the content of the file. Only the data-blocks are marked free. At this stage if the data blocks are re-reads somehow, it WILL be possible to recover the deleted content!

However, if the data block just marked as free is overwritten by some new file, then it may not be possible to recover the complete file. Now it depends on the file-system's algorithm of selecting free blocks, which block is taken for writing new files.

The Slueth Kit

The Slueth Kit is a compilation of a set of binaries that come in very handy for reading filesystems. For better usability the commands have been categorized based on the *layers* they work on. The utilities being aimed at forensic investigation, work directly on the device. All the utilities take the device file name as an argument. One can also use – as its used often – these binaries work on files that contain disk images. Also all commands accept the file-system type as an argument.

Content/Data Layer: This layer corresponds to the data-blocks or clusters. Commands that can read/query this layer start with `d`

- dcat : It displays contents of disk blocks on the stdout. One can specify the block number and get it *catted* on the screen.
- dls : lists contents of deleted blocks(umrm in TCT)
- dcalc : maps between dd images and dls
- dstat : lists statistics associated with specific disk blocks

File System Layer: All binaries start with `fs`

- fsstat : Displays details(certain statistics) about file-system

Meta-Data Layer: All the commands under this category interact with the inode of the file-system. For the same reason, all names start with `i`

- ils : displays inode details.
- istat : displays information abt specific inodes
- icat : cats information stored on the blocks allocated to an inode. This can be used to recover files from a disk without even mounting it!
- ifind : search for some inode from an image, based on the block/content number or filename etc. ifind does not accept filenames with regular expressions.

Human Interface Layer: Start with `f`. This one starts with f because Human Interface from the file-system point of view is the file name.

- fls : List files just like `ls`
- ffind : determine which file has allocated an inode on an image

Media Management Layer: starts WITH `mm`

- mmls : displays list of partitions or disk slices etc.

Other tools...

hash database tools,
the `file` tool and `sorter`
timeline tools
eg fls -f fat12 /dev/fd0

The above command will list the contents of the floppy, if there is one in the drive

Command used to generate MAC Times

```
fls -f fat12 -l -zMST -s45000 fl-260404-RJL1.img | sed  
's/)<CTRL+V,tab>/)\n<CTRL+V,tab>/g'
```

fls: lists the files from the image.

options used:

- f fat12 : Tells the program that the image, fl-260404-RJL1.img is of a disk formatted with FAT12 file-system
- l : Give a long listing output. So that we can get the size of the file
- z MST : Use MST as the time-zone specification string.
- s 45000 : Skew the time-stamp readings by 45000 seconds.

The output of this command comes something like this:

```
r/r * 5: CamShell.dll (_AMSHHELL.DLL) 2001.02.03 19:44:16 (IST)  
2004.04.26 00:00:00 (IST) 2004.04.26 09:46:18 (IST) 36864 0
```

This was piped to a sed(1). The sed(1) command,

's/)<CTRL+V,tab>/)\n<CTRL+V,tab>/g' replaces presence of a closing parenthesis and a tab character with a closing parenthesis, a new-line character and a tab-character. This was done merely to increase the readability of the output. Since tab is a special character interpreted by the shell, it has to be typed specially using the escape sequence CTRL+V.

Linux commands

dcfldd(1), nc(1), strings(1), grep(1), mount(8), md5sum(1), ls(1), gunzip(1), head(1), tail(1), cat(1), diff(1), file(1), KHexEdit and certainly bash(1)

nc (1) - arbitrary TCP and UDP connections and listens. This utility is called netcat, and can be used as a simple client server pair to perform any data transfer.

strings (1) - print the strings of printable characters in files. Very useful to analyze binary files.

grep (1) - print lines matching a pattern.

mount (8) - mount a file system

md5sum (1) - compute and check MD5 message digest. It calculates the md5 cryptographic hash

ls (1) - list directory contents

head (1) - output the first part of files. One can use the -c option to output only first few bytes

tail (1) - output the last part of files

cat (1) - concatenate files and print on the standard output

diff (1) - find differences between two files

file (1) - determine file type

bash (1) - GNU Bourne-Again Shell

sed (1) - Sed is a stream editor. One can write substitution commands

to edit the stream passing through the editor.

stat (1) - display file or filesystem status. The command shows all details regarding the file including its mac time stamps.

gunzip [gzip] (1) - compress or expand files

© SANS Institute 2000 - 2005, Author retains full rights.

APPENDIX B

CRACKING CAMOUFLAGE PASSWORD

Camouflage is a software used to hide one file into another. This is called Steganography. Using Camouflage, one can embed any number of files into another file. eg., one can embed *dirty-snaps.zip* and *last-night.jpg* into another file called *MyResume.doc* in order to hide them, eventually making them inaccessible. After embedding the files, *MyResume.doc* (called the *carrier*) will still retain all its earlier capabilities as a MS Word file. The data is *appended* at the tail of the document, thereby avoiding any problems the application might face. Moreover the data is also scrambled so that it could not be detected using techniques employed by tools like *lazarus* and *sorter*. The software also lets the user protect the embedded files using a password.

The fact that Camouflage stores password in the carrier itself may sound foolish at first, but, there is more to it. Camouflage stores it into the carrier, but not before scrambling the password string. Moreover the scrambling generates a string that is complete binary, eventually saving the password from being detected by a simple analysis using strings(1).

Camouflage stores the scrambled password at an offset exactly 275 bytes before the end of file. The scrambled password is of the same length as the original. The simple technique used by *Guillermi* lets detect this.

A file is *Camouflaged*, using a password "aaaa". It was found that the bytes at location 275 from the tail were getting modified. Moreover it also showed that the first four bytes remained the same when the file was *Camouflaged* using a password containing more a's. This indicated that there was some hash that was used to scramble the password. The hash was weak enough, that it would generate the same result per byte. Its important to note here that the hash was applied at each byte. This indicated that the hash would be nothing but some binary operation. The first thing that comes to mind is XOR. Mainly because, its simple, easy to implement, and equally easy to recover. By trying a long password containing some 100 a's it was possible to detect the hash used. Camouflage uses the XOR technique to store passwords. This technique was cracked by *Guillermi*[5]. The XOR hash string so recovered is as follows:

```
0295 7A22 0CA6 14E1 E1CF BF65 206F 9EB3 9965 4A53 FBF6
7554 AD23 CD7E 9C29 E7FC E2F9 4DD2 424E 06C0 F89A 1C62
3874 2400 55DF 41CB 01A2 B7F3 8F8A DDAC 3383 6029 F378
243E 7AEB D3E4 9D9D 4394 4AC7 456D 2574 EB0B 98C9 7CFC
C8BA 326B 00D3 C5C2 9434 AFB0 E595 7D2A 84A4 5FE5 6E27
2ADB 967E 3E48 3946 CF6F 71AA 3C31 9AA9 9E8F 8973 B339
CA32 D5F0 3159 7C02 2E86 37F9 2B7E 51F2 4181 0CD4 6515
F770 D419 9820 BF20 B855 67CC 8118 8C13 3C63 3C92 11E4
5B1B 0822 604C 4AC5 8AB3 C575 C390 7AF2 B2B6 C8D0 388A
C286 F0AC E9CA 5C4E 3E09 2978 2999 5A84 D5BA 5ED5 927A
38FA D060 ECF5 27BA EEB7 DE9F 9BDE 65D4 7639 769C DA68
8DA8 A0A6 1ED9 DB0F 4DAB 92CD 71
```

APPENDIX C

CHAIN OF CUSTODY

Evidence custody form

Case: Draupadi Hack

Chain of Custody

1.	Forensic Team Members	Pramod S. Pawar Nihar S. Khedekar Vijay Kumar VK
2.	Description of Evidence	Nine hard disk partition images of the compromised server Draupadi were obtained.
3.	Person receiving Evidence	Nihar S. Khedekar
4.	Case No.	Aug-DH-040815
5.	Hash values of the evidence	661a4f317ce620e2f49de820a5d04257 sda1-dd 6b7bbf152e11e6f346357dc42c838d89 sda10-dd 22b2939c417e2f0333bf41dde891ebbf sda5-dd 56a125d04fa2ea3beb9c355921ef9bda sda7-dd cba7fada45bcaa8d0402cdd7d484c10b sda8-dd debf77cc75c0e48ceb1274f9160d3abc sda9-dd b2ec6a068f2c57495a9ad39f1223c60d sdb1-dd fe3df9d054d76fef3d038d1d604256b sdc1-dd cfa9ce8308700f2ebfdef2424445a3cc sdc6-dd

Sr. No	Date/Time	Release by	Received by
1.	Date: 15 th August 2004	Mr. K Arjuna, System Administrator, Software Techniques Ltd., Bangalore	Nihar S. Khedekar CDAC, Bangalore
	Time: 17:30:00 IST	Sd/-	Sd/-