Digital Forensics Research Workshop Challenge 2009

Wouter S. van Dongen (wouter.vandongen@os3.nl)
Alain van Hoof (alain.vanhoof@os3.nl)

July 2009

System & Network Engineering

University of Amsterdam
System & Network engineering

Student Research Project 2 report
`http://www.os3.nl/rp`

Contents

1 Introduction **4**
1.1 Challenge details 4
1.2 Overview of the available challenge data 5
1.3 Challenge participation 6

2 Methodology **7**
2.1 Tools and applications 7
2.2 Time zones 8
2.3 Linux time stamps 9
2.4 SSH login traces in files 10
2.4.1 auth.log 12
2.4.2 ConsoleKit 12
2.4.3 wtmp 13
2.4.4 known_hosts files 13
2.5 Booting the linux-images on a Playstation 3 13

3 Investigation **15**
3.1 Accounts 15
3.2 Recovery of deleted files 16
3.2.1 The recovery 17
3.3 Time stamps and mount options 19
3.3.1 Checking mount options 20
3.3.2 Experimentally determining mount options 20
3.4 Mardi Gras pictures 22
3.4.1 location and description 22
3.4.2 Related traces 24
3.5 Drug recipes 24
3.5.1 Location and description 24
3.5.2 Related traces 25
3.6 Backdoor 26
3.7 Location and description 26
3.7.1 Related traces 27
3.8 Thumb drive 27
3.9 Network traffic 30
3.9.1 Network time stamps 30
3.9.2 MAC and IP addresses 30
3.9.3 Playstation and PSP Traffic 31
3.9.4 Playstation home 31
3.9.5 Mardigrass related traffic 32
3.9.6 SSH network traffic 32
3.9.7 known_hosts files 34
3.9.8 backdoor traffic 34
3.10 Application history 35
3.10.1 Firefox 35
3.10.2 Opera 36
3.11 Putting the gathered information together 37

4 Conclusion **38**
4.1 Relevant user activity 38
4.2 Drug recipes 38
4.3 Mardi Gras pictures 38
4.4 Backdoor 39
4.5 Relation between suspect and JHUISI Playstation 39

5 Remarks **40**

6 Further research **40**

7 Acknowledgments **40**

8 Bibliography **41**

9 Appendix A - Overview **42**

1 Introduction

The Digital Forensics Research Workshop (DFRWS) is a nonprofit, volunteer organization dedicated to the sharing of knowledge and ideas about digital forensics research. DFRWS organizes annual challenges to help drive the direction of research and development. The DFRWS challenge is well known among forensics institutes and has lead to many new techniques and tools for digital forensic investigations.

This year's challenge focuses on a system architecture (IBM 64bit PowerPC) that is not frequently encountered in digital forensics. Furthermore, the involved operating system (Linux) and the ext3 file system are is not as widely supported in forensic tools as for example Microsoft Windows and NTFS. This year's challenge is described as follows:

> *The DFRWS 2009 Challenge focuses on the development of tools and techniques for analyzing Playstation 3's (PS3s). The Playstation 3 is a powerful, Cell processor-based system that can run both its native OS (which has significant DRM features that also thwart forensic investigation) and modern versions of Linux. This challenge focuses on the Linux and network aspects of PS3s, and does not touch the DRM protected data. The challenge scenario requires analysis of a physical memory dump, filesystem images, and network traces involving 2 PS3's and a Playstation Portable (PSP).*

Information about the DFRWS challenge and the corresponding data can be found on the website of the DFRWS[1].

1.1 Challenge details

The challenge details are presented on the website of the DFRWS. The challenge is described as follows:

> In early 2009 it came to the attention of investigators that an individual with the nickname "nssal" was using a Sony Playstation3 (PS3) to make illicit images (specifically, certain images depicting Mardi Gras activities) available to other PS3 users. Investigators determined that "nssal" was connecting from an IP address in New Orleans, and they began capturing network traffic with the goal of catching "nssal" red-handed. Based on their initial surveillance, it appeared that "nssal" had advanced knowledge of Linux and digital forensics.
>
> On March 11, 2009 investigators observed "nssal" communicating with another PS3 user and exchanging unknown data. With proper legal authorization, the investigators entered the suspects premises and found him in front of a PS3 that was running Linux. They interviewed the suspect and determined that he was a digital forensics researcher who was developing memory acquisition and analysis tools for Linux on the PS3. He denied having exchanged any information with other PS3 users.
>
> Investigators captured physical memory of the Linux system on the PS3 using tools found on the system. This physical memory dump is present in the file nssal-physicalmem.dd.bz2. Investigators also acquired a forensic duplicate of the Linux partition on the PS3 (present in file nssal-linux-side-fs.dd.bz2) and the suspects thumb-drive (present in file nssal-thumb-fs.dd.bz2). Several network traces are also available. The first network trace is based on early surveillance of the suspect; this network trace is named nssal-capture-1.pcap.bz2. A second trace, nssal-capture-2.pcap.bz2,

[1] `http://dfrws.org/2009/challenge/`

contains communication between ”nssal” and another machine located at Johns Hopkins University. The network administrator in the lab at Johns Hopkins identified the machine as another PS3. This administrator regularly monitors communication and was able to provide a third network trace, jhuisi-capture-1.pcap.bz2, which contains traffic transmitted between the ”nssal” PS3 and the PS3 in the Johns Hopkins lab. The system administrator also obtained a filesystem image of the PS3 at Johns Hopkins (present in jhuisi-linux-side-fs.dd.bz2) but was unable to obtain a physical memory dump.

You have been asked to assist investigators with the following questions:

- *What relevant user activity can be reconstructed from the available forensic data and what does it show?*
- *Is there evidence of inappropriate or suspicious activity on the system?*
- *Is there evidence of collaboration with an outside party? If so, what can be determined about the identity of the outside party? How was any collaboration conducted?*
- *Is there evidence that illicit data (specifically, Mardi Gras images) was exchanged? If so, what can be determined about that data and the manner of transfer?*
- *What data (if any) was provided by the Johns Hopkins PS3?*
- *The suspect claims that he was not responsible for any transfer of data. What evidence do you have to show that remote, unauthorized access to the system might have occurred, and does this evidence exonerate the suspect?*

1.2 Overview of the available challenge data

As described in section 1.1, different types of data are available for investigation. Table 1 summarizes the available data and includes a calculated MD5 hash (after uncompressing). Figure 1 depicts the challenge scenario and adds technical information to the hardware described in the challenge details.

nssal-physicalmem.dd	MD5: `5ffa6839dbab169c9c44436c80f8ea9b` Physical memory dump of the PS3 of the suspect
nssal-linux-side-fs.dd	MD5: `a64f017400123b7eff2c5868f8e784e1` Forensic duplicate of the Linux partition on the PS3 of the suspect
nssal-thumb-fs.dd	MD5: `d4a8ff499355d82c9df76254dcb0cb1d` Forensic duplicate of the suspect’s thumb drive
jhuisi-linux-side-fs.dd	MD5: `3ee30c25e5bca832b93ddf9eee88cb1b` Forensic duplicate of the Linux partition of the PS3 at Johns Hopkins University
nssal-capture-1.pcap	MD5: `e6ba905d4a0630915600b00ab9712499` Network trace based on early surveillance of the suspect
nssal-capture-2.pcap	MD5: `fd28ae8fff1430b19ceec9785c2b027b` Network trace based on surveillance of the suspect
jhuisi-capture-1.pcap	MD5: `9bb90b7e6c5b8d7401b7621969017b01` Network trace provided by the Johns Hopkins University

Table 1: The challenge data available and used for investigation.

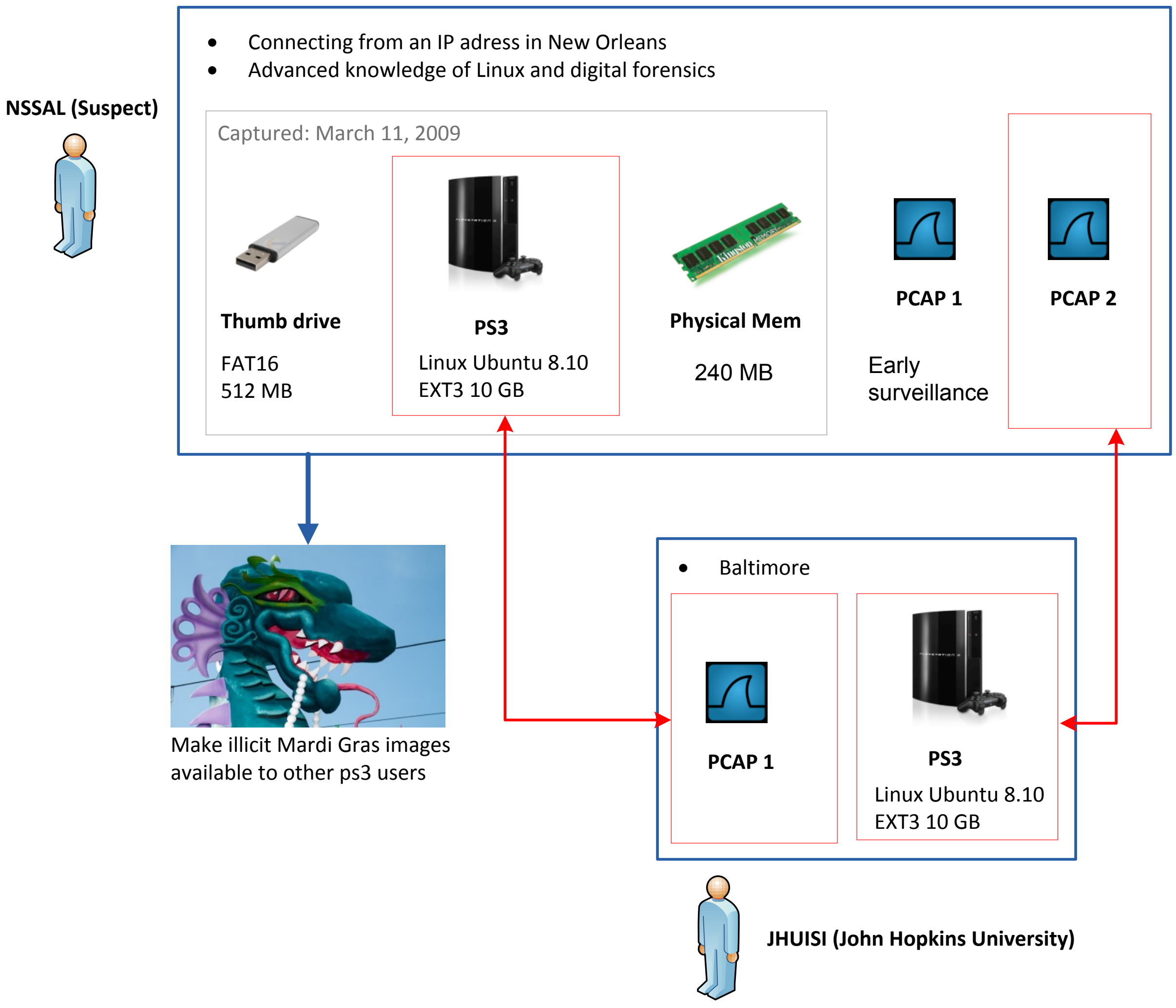


Figure 1: Scenario

1.3 Challenge participation

This report is created as part of a five week research project done as part of the System- and Network-engineering Master education at the University of Amsterdam. The research done in this report is not solely related to answering the questions set-up by the DFRWS challenge. This report provides additional research for into Linux time stamps and is written in such a way that it can be used as a reference in other Linux investigations.

2 Methodology

2.1 Tools and applications

Linux

Standard Linux commands and additional Linux utilities will be used to examine the Linux file system images. The utilities and commands will be executed on either the raw images or on read-only mounted images. The read-only mounts will be done as follows:

```
# Mount option as described by the mount manual:
# -r; mount image read-only
# nodev; do not interpret character or block special devices on the file system.
# noexec; do not allow direct execution of any binaries on the mounted file system.
# nosuid; do not allow set-user-identifier or set-group-identifier bits to take effect.

mount -r <image> <target> -o loop,nodev,noexec,nosuid
```

X-Ways Forensics

X-Ways Forensics[2] will be used to conduct keyword searches, to analyze files on a byte level and to quickly browse through all images.

Wireshark

The available network captures will be analyzed using Wireshark 1.2 [3]. Wireshark is a network protocol analyzer with advanced features to examine, filter, convert and follow network traffic/streams.

Aftertime and Microsoft Excel

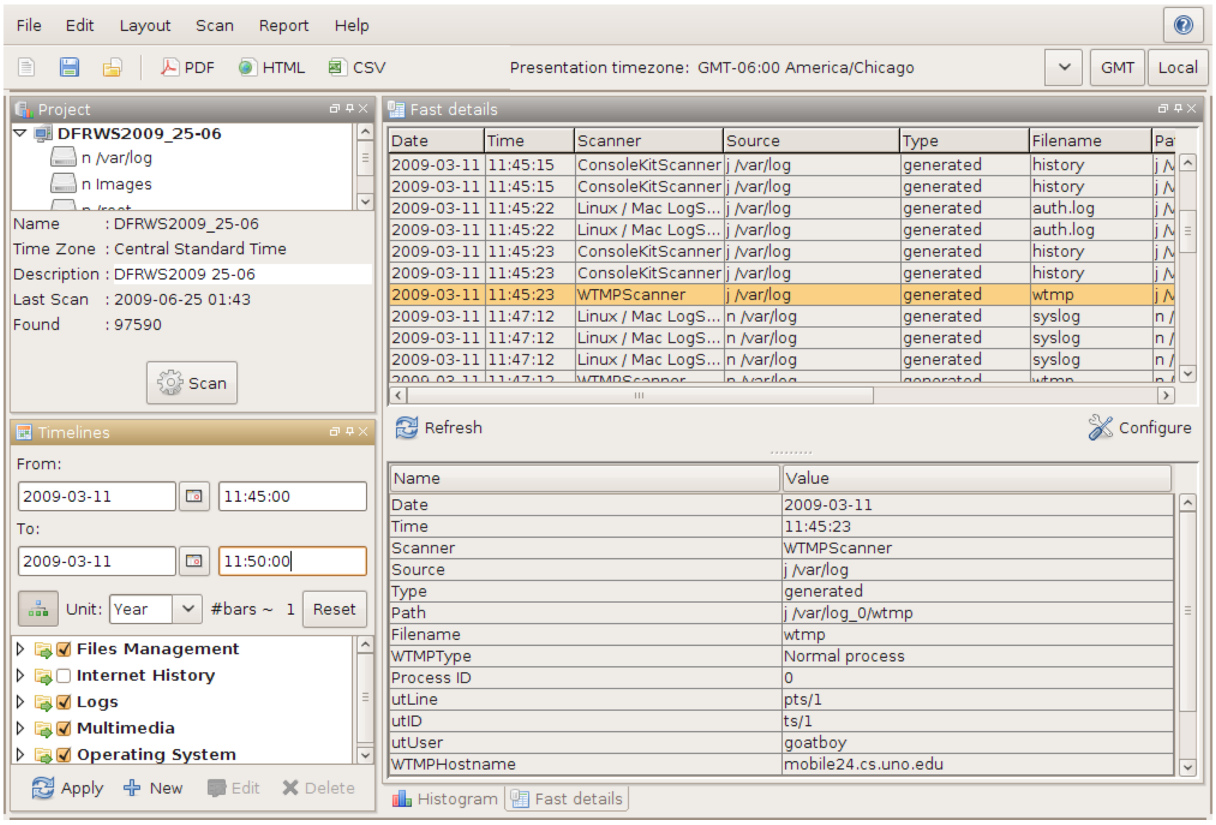
Correlating events between the two Playstation systems will be an important part of the investigation. Combining timed data from within the various sources can be a complex task. A combination of Aftertime and Microsoft Excel will be used to accomplish this task. Aftertime is an application developed by the Netherlands Forensic Institute (NFI) to assist in investigations concerning time stamps located in different types of media. Aftertime has the ability to collect time stamps, convert them to a common format and display them in such a way that they can be useful in an investigation. This way it is possible to view all events stored with different time stamp formats and in different time zones in the correct order taking daylight savings into account. Aftertime is able to parse Microsoft Windows event logs and to some extent Mac OS X log files. On our request and with our input Aftertime has been extended by the NFI to include Linux log files. Figure 2 provides a screenshot of Aftertime.

All events will be exported to a CSV[4] file with Aftertime. Microsoft Excel will be used to quickly sort, filter and correlate events which is illustrated in Figure 3.

[2]X-Ways Forensics is available from: `http://www.x-ways.net/`

[3]Wireshark is available from: `http://www.wireshark.org/`

[4]CSV: Comma-Separated Values format. A text file with column values (not formulas) separated by a comma and lines representing a row, useful for exchanging data among different applications.



Name : DFRWS2009_25-06
Time Zone : Central Standard Time
Description : DFRWS2009 25-06
Last Scan : 2009-06-25 01:43
Found : 97590

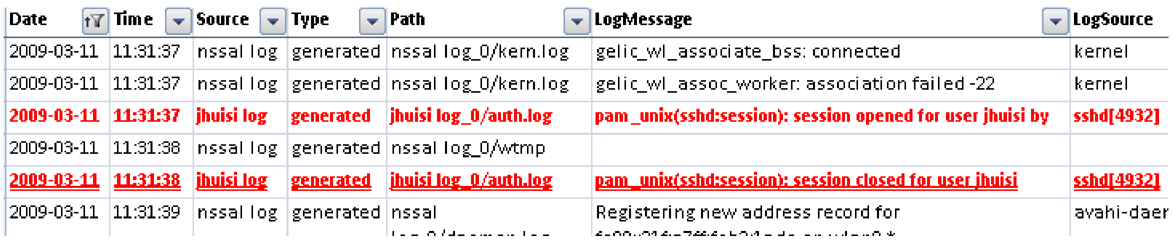
From: 2009-03-11 11:45:00
To: 2009-03-11 11:50:00
Unit: Year #bars ~ 1 Reset

- Files Management [✓]
- Internet History [ ]
- Logs [✓]
- Multimedia [✓]
- Operating System [✓]

Date	Time	Scanner	Source	Type	Filename	Pa
2009-03-11	11:45:15	ConsoleKitScanner	j /var/log	generated	history	j /\
2009-03-11	11:45:15	ConsoleKitScanner	j /var/log	generated	history	j /\
2009-03-11	11:45:22	Linux / Mac LogS...	j /var/log	generated	auth.log	j /\
2009-03-11	11:45:22	Linux / Mac LogS...	j /var/log	generated	auth.log	j /\
2009-03-11	11:45:23	ConsoleKitScanner	j /var/log	generated	history	j /\
2009-03-11	11:45:23	ConsoleKitScanner	j /var/log	generated	history	j /\
2009-03-11	11:45:23	WTMPScanner	j /var/log	generated	wtmp	j /\
2009-03-11	11:47:12	Linux / Mac LogS...	n /var/log	generated	syslog	n /
2009-03-11	11:47:12	Linux / Mac LogS...	n /var/log	generated	syslog	n /
2009-03-11	11:47:12	Linux / Mac LogS...	n /var/log	generated	syslog	n /

Name	Value
Date	2009-03-11
Time	11:45:23
Scanner	WTMPScanner
Source	j /var/log
Type	generated
Path	j /var/log_0/wtmp
Filename	wtmp
WTMPType	Normal process
Process ID	0
utLine	pts/1
utID	ts/1
utUser	goatboy
WTMPHostname	mobile24.cs.uno.edu

Figure 2: Screenshot of Aftertime.



Date	Time	Source	Type	Path	LogMessage	LogSource
2009-03-11	11:31:37	nssal log	generated	nssal log_0/kern.log	gelic_wl_associate_bss: connected	kernel
2009-03-11	11:31:37	nssal log	generated	nssal log_0/kern.log	gelic_wl_assoc_worker: association failed -22	kernel
2009-03-11	**11:31:37**	**jhuisi log**	**generated**	**jhuisi log_0/auth.log**	**pam_unix(sshd:session): session opened for user jhuisi by**	**sshd[4932]**
2009-03-11	11:31:38	nssal log	generated	nssal log_0/wtmp		
2009-03-11	**11:31:38**	**jhuisi log**	**generated**	**jhuisi log_0/auth.log**	**pam_unix(sshd:session): session closed for user jhuisi**	**sshd[4932]**
2009-03-11	11:31:39	nssal log	generated	nssal	Registering new address record for	avahi-daer

Figure 3: Screenshot of Microsoft Excel with exported Aftertime data.

2.2 Time zones

The evidence is acquired from systems in different time zones[5]. To present the evidence, correct interpretation of the time is of great importance. On a Linux-system the file `/etc/timezone` is used to store the local time of the system. Table 2 displays relevant time zone information. Daylights saving time begins the second sunday in March (March 8th 2009) and ends first sunday of November (November 1st 2009) for both time zones.

To obtain the view of the suspect and investigators asking the questions, time stamps in this report will be examined and presented in Central Standard Time with Daylights saving enabled. The dates in this report are formatted in YYYY-MM-DD to avoid confusion on the month and the day.

[5]From `http://wordnet.princeton.edu/`: Regions of the globe (loosely divided by longitude) throughout which the same standard time is used

Disk Image	**Physical Location**	`/etc/timezone`	**Code**	**UTC Offset Daylight Saving**	**UTC Offset**
nssal-linux-fs	New Orleans	America/Chicago	CDT	-5	-6
jhuis-linux-fs	Baltimore	US/Eastern	EST	-4	-5

Table 2: Time zones and `/etc/timezone` of the acquired evidence filesystem images

2.3 Linux time stamps

In order to determine when files are accessed, modified or created, the Linux time stamps of files will be of great importance. The Linux ext3 file system registers several time stamps of a file. The access time is updated when reading a file or directory, the modify time is updated when the contents of a file or directory changes and the change time is updated when the meta-data of a file or directory changes. The 'stat' command shows is able to show these time stamps of a file:

```
~/Images$ stat 3316820191_4737c3edf4.jpg
  File: '3316820191_4737c3edf4.jpg'
  Size: 135809            Blocks: 280        IO Block: 4096   regular file
Device: fe01h/65025d        Inode: 504801      Links: 1
Access: (0755/-rwxr-xr-x)  Uid: ( 1000/   nssal)   Gid: ( 1000/   nssal)
Access: 2009-03-06 15:21:32.000000000 -0600
Modify: 2009-03-02 08:34:50.000000000 -0600
Change: 2009-03-06 15:21:31.000000000 -0600
```

The above described behaviour is observed on various UNIX-type operating systems including the Linux operating systems. In the book "File System Forensic Analysis" [1], Carrier describes the forensic information that can be extracted from the time stamps information of the ext3 file system. He states that the 'touch' command can be used to modify the access time and/or modify time by a user on the system. However, during this investigation another important issue was found in the Linux operating system that is not addressed in well known literature such as "File System Forensic Analysis" [1]. The mount options used to mount an ext3 file system can be of great importance to the forensic use of the access, modify and change time stamps. Table 3 gives an overview when time stamps of the ext3 file system are updated.

Action	**Updated time stamp(s)**		
Creation	Access	Modify	Change
Read	Access*		
Modify Content		Modify	Change
Copy (source)	Access*		
Copy (target does not exist)	Access	Modify	Change
Copy (target does exist)	**	Modify	Change
Move (target does not exist)			Change
Move (target does exist)			Change

* This behaviour depends on the mount options of the Linux ext3 filesystem.
** The time stamp of the file which is over written is used.

Table 3: Default ext3 file system behaviour for file and directory time stamps.

Most of the actions result in the update of one or more time stamps. Depending on the mount options the access time is always updated ('atime'), never updated ('noatime') or updated when the access time is less or equal modify or change time ('relatime'). When a file is moved to an other filename that already exist the access time of the original file is preserved.

2.4 SSH login traces in files

SSH is the most commonly used method to access remote Linux systems. Besides this, SSH can also be used to exchange files by using SFTP[6] or SCP[7].

Multiple log files will be examined on both Linux images to determine if remote access with SSH took place. To be able to find traces of remote access to a system, it is important to know where to look for traces, if traces are related and what their relationship is. An SSH session is started with the 'ssh <username>@<hostname>' command to login to a remote system or the command 'scp <username>@<hostname>:<filename> <filename>' to copy a file from a host or vice versa. Besides this the command 'sftp <username>@<hostname>' is used to start a SFTP session. When the 'ssh','scp' or 'sftp' is used (from now on referenced to as an SSH session), log files on both the local and the remote system are updated. It is not possible to distinguish a 'ssh' session from a 'scp' session by using the log files. 'sftp' sessions can be distinguished from 'ssh' and 'scp' sessions by using the `auth.log` file which shows the message 'subsystem request for sftp' after the 'session opend' message, see section 2.4.1.

Figure 4 shows which files are updated. Table 4 shows examples of SSH traces found in log files corresponding to the colored numbers in Figure 4.

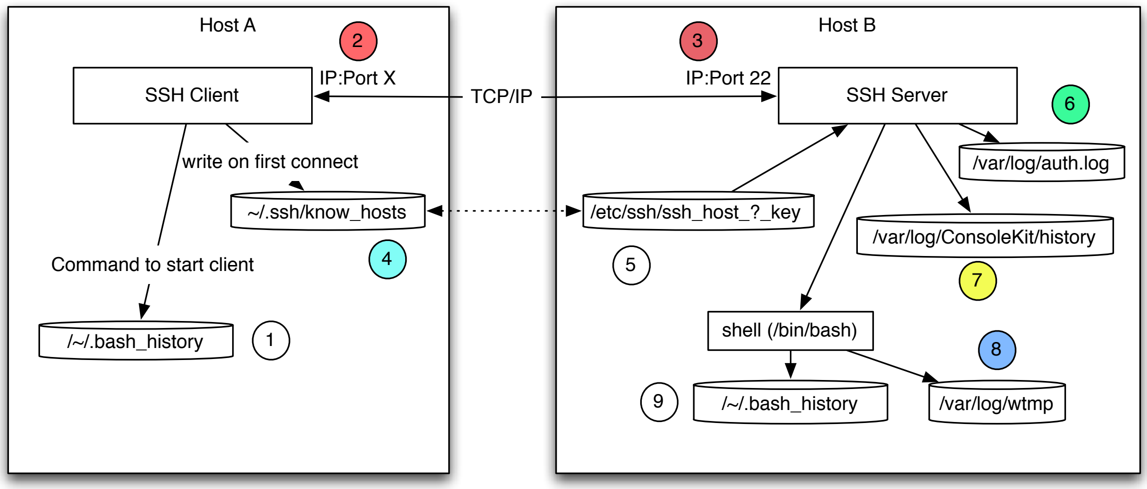


Figure 4: SSH Traces in files

[6]In computing, the SSH File Transfer Protocol (sometimes called Secure File Transfer Protocol or SFTP) is a network protocol that provides file transfer and manipulation functionality over any reliable data stream. It is typically used with version two of the SSH protocol (TCP port 22) to provide secure file transfer, but is intended to be usable with other protocols as well [7].

[7]Secure Copy or SCP is a means of securely transferring computer files between a local and a remote host or between two remote hosts, using the Secure Shell (SSH) protocol [6].

1	Local `.bash_history` jhuisi-linux-side-fs.dd	ssh jhuisi@137.30.123.40* or scp jhuisi@137.30.123.40*
2 & 3	Network Trace nssal-capture-2.pcap	2009-03-11 11:49:37 128.220.249.83 137.30.123.40 TCP 51874 > ssh [SYN] 2009-03-11 11:49:37 137.30.123.40 128.220.249.83 TCP ssh > 51874 [SYN, ACK] 2009-03-11 11:49:37 128.220.249.83 137.30.123.40 TCP 51874 > ssh [ACK]
4	known_hosts (7) jhuisi-linux-side-fs.dd	stat /home/jhuisi/.ssh/known_hosts Access: 2009-03-11 11:51:57.000000000 -0500 Modify: 2009-03-11 11:49:40.000000000 -0500 Change: 2009-03-11 11:49:40.000000000 -0500
6	auth.log nssal-linux-side-fs.dd	Mar 11 11:49:45 nssal-ps3 sshd[3208]: Accepted password for jhuisi from 128.220.249.83 port 51874 ssh2 Mar 11 11:49:45 nssal-ps3 sshd[3208]: pam_unix(sshd:session): session opened for user jhuisi by (uid=0)

7	ConsoleKit/history
	nssal-linux-side-fs.dd
		1236790186.127 type=SEAT_SESSION_ADDED : seat-id='Seat2'
		session-id='Session2' session-type='' session-x11-display=''
		session-x11-display-device='' session-display-device='/dev/ssh'
		session-remote-host-name='128.220.249.83' session-is-local=FALSE
		session-unix-user=1001 session-creation-time='2009-03-11T16:49:45.880532Z'

8	wtmp nssal-linux-side-fs.dd	jhuisi pts/2 128.220.249.83 Wed Mar 11 11:49 - 12:09 (00:20)
9	Remote `.bash_history` nssal-linux-side-fs.dd	Standard Linux commands

* Trace was not found due to in readable format of bash history, but the entry illustrates what could have been found.

Table 4: An example SSH session to host 137.30.123.40 and the traces found in log files.

The `.bash_history` on the system initiating the SSH session (referenced as number 1 in Figure 4) will show the start of the session as previously described. Commands executed during an 'ssh' session (not 'scp' and 'sftp') on the remote system are recorded in the `.bash_history` of the <username> (number 9 in Figure 4) on the remote host.

An SSH session found in the network traffic capture `nssal-network-capture-2.pcap` and the entries in the log files can be used to correlate events between both systems (Figure 4 dots number 2 & 3). This is illustrated in Table 4. The Wireshark output of the example SSH session in the network traffic is shown in the first row of Table 4. The time stamp, IP addresses and the port numbers (51874 & ssh=22) are values in the data that can be used for correlation. The port number 51874 used to initiate the SSH session in this example, can be found in `auth.log` as described in the next sub-section. Besides `auth.log`, the other log file traces numbered in Figure 4 will be explained in separate sub-sections of this section.

2.4.1 auth.log

The `auth.log` file (number 4 in Figure 4) typically contains logging of the 'sudo' command[8] activity, creation of user accounts and logging of the Pluggable Authentication Module[9] (PAM). PAM does not only log local logins but also remote logins sessions created with SSH for example.

The default Ubuntu 8.10 system log configuration has the following line in its configuration file `/etc/syslog.conf`.

```
auth,authpriv.*               /var/log/auth.log
```

This line generates the contents of `/var/log/auth.log`. To make sure `auth.log` is updated, checking for presence of this line in `/etc/syslog.conf` needs to be done.

The log rotation facilities of a Linux Ubuntu system archives the `auth.log` file as `auth.log.0` (uncompressed), `auth.log.1.gz` , `auth.log.2.gz` etc. (compressed), this is done big avoid big log files.

As an example, the command 'zgrep jhuisi auth.log*'[10] is executed in the `/var/log` directory of the `nssal-linux-side-fs.dd` image. SSH session entries containing the text 'jhuisi' were found. An illustrative part of these entries:

```
       auth.log:
Mar 11 11:49:45 nssal-ps3 sshd[3208]: Accepted password for jhuisi from 128.220.249.83 port 51874 ssh2
Mar 11 11:49:45 nssal-ps3 sshd[3208]: pam_unix(sshd:session): session opened for user jhuisi by (uid=0)
Mar 11 12:09:47 nssal-ps3 sshd[3208]: pam_unix(sshd:session): session closed for user jhuisi
```

The first part of the lines show a time stamp (without year), the host name (nssal-ps3) where the log entry was generated and the process that generated the message (sshd) including its process number (3208). The process and process number can be used to uniquely identify the SSH session within the `auth.log` file. The second part of the lines shows additional information about the session. The password of the user 'jhuisi' was accepted for remote host IP 128.220.249.83 on port 51874 and the start and end of the session are indicated.

As a second example, the command 'zgrep goatboy auth.log*' in the `/var/log` directory of the `jhuisi-linux-side-fs.dd` image was executed. Entries containing the text 'goatboy' were returned. A failed and a successful SSH login are shown in this example output:

```
    auth.log:
Mar 11 12:32:25 ps3 sshd[4972]: Accepted password for goatboy from 128.220.251.228 port 9477 ssh2
Mar 11 12:32:25 ps3 sshd[4972]: pam_unix(sshd:session): session opened for  user goatboy by (uid=0)
Mar 11 12:32:42 ps3 sshd[4972]: pam_unix(sshd:session): session closed for user goatboy
Mar 11 12:42:34 ps3 sshd[5232]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0
                                tty=ssh ruser= rhost=mobile24.cs.uno.edu  user=goatboy
Mar 11 12:42:37 ps3 sshd[5232]: Failed password for goatboy from 137.30.123.40 port 35892 ssh2
Mar 11 12:42:50 ps3 sshd[5232]: Failed password for goatboy from 137.30.123.40 port 35892 ssh2
```

In this second example two sessions are shown, each uniquely identified by the process and process id. Besides a successful session (sshd[4472]), an unsuccessful sessions can be identified (sshd[5232]) including from what IP address or host name (see 'rhost=') these sessions were initialized.

And again, note the `auth.log` file like most other files in `/var/log` contains time stamps where the year is absent.

2.4.2 ConsoleKit

On a Ubuntu 8.10 system the ConsoleKit service is used to assign users to a keyboard and a mouse, this allows for switching the active user in a graphical environment for example. To be able to have "complete" control the ConsoleKit keeps track of all access to a system, including remote SSH logins or SFTP sessions. The logging of the ConsoleKit is done in the file

[8]sudo allows a normal user to become root (the UNIX super user)

[9]PAM is an extensible part of a Linux/UNIX system and authenticates users logging in.

[10]zgrep finds a string in compressed and uncompressed files

`/var/log/ConsoleKit/history` in Figure 4 at number 5. Because all access is logged, traces of SSH/SFTP sessions can be found and investigated. The following example shows the start of an SSH session in `ConsoleKit/history` :

```
1231994310.230 type=SEAT_SESSION_ADDED : seat-id='Seat2' session-id='Session11' session-type=''
session-x11-display='' session-x11-display-device='' session-display-device='/dev/ssh'
session-remote-host-name='mobile183.cs.uno.edu' session-is-local=FALSE session-unix-user=1000
session-creation-time='2009-01-15T04:38:30.194625Z'
```

Information about the SSH session can be found in this line. The first number is the creation date of entry in `ConsoleKit/history`, formatted in the form of a UNIX epoch time stamp[11]. All other parts of the entry are in the form parameter = value. 'session-id' and 'session-creation-time' uniquely identify the session, 'session-display-device' indicates a SSH session from host 'session-remote-host-name' by user with UID[12] 'session-unix-user'.

2.4.3 wtmp

On a Unix (Linux) system a login/logout of an account and a system reboot is stored in `/var/log/wtmp`. The `wtmp` file is in binary format and can be read using the command 'last'. The log rotation facilities of Ubuntu 8.10 archives the `/var/log/wtmp` file as `/var/log/wtmp.1` on a monthly basis. The command 'last -f <filename>' can be used to access the archived files.

When using an x86 based Linux system to view the contents of the wtmp file of the `nssal-linux-side-fs.dd` image (PPC64) using 'last -f' the entries were not displayed correctly:

```
reboot   system boot  2.6.28          Fri Sep 11 00:38 - 05:34 (3931+04:56)
nssal    pts/1                        Wed Sep 19 05:28 - crash (-16115+-11:
nssal    pts/0                        Thu Dec  2 15:47 - crash (-10674+-15:
```

After booting the `nssal-linux-side-fs.dd` image on a PS3 the 'last' command was executed, showing the correct wtmp information. Looking at the binary data in the `wtmp` file at offset 340 (total entry size 384 bytes) which contains a 4 bytes UNIX time stamp and comparing these values with a wtmp file created on an x86 Linux system an big endian against little endian issue was discovered. A simple python script was written to convert the PPC64 (big endian) wtmp to x86 (little endian) and back again. Now the output looks normal on an x86 based Linux system:

```
reboot   system boot  2.6.28          Wed Mar 11 12:43 - 06:29 (96+17:46)
nssal    pts/1                        Wed Mar 11 12:36 - crash  (00:07)
nssal    pts/0                        Wed Mar 11 12:14 - crash  (00:28)
```

2.4.4 known_hosts files

The public key of the remote host is placed in the file `~/.ssh/known_hosts` on the local host when an SSH connection is made for the first time or when the remote host has a new public key. The `known_hosts` file is only updated/created when the SSH configuration file `/etc/ssh/sshd_config` does not contain '`IgnoreUserKnownHosts yes`'. The public key of a host can be found in the file `/etc/ssh/ssh_host_rsa_key.pub`.

2.5 Booting the linux-images on a Playstation 3

To observe the behaviour and "look and feel" of a Linux system on a Playstation 3 and in particular the Linux file system images provided by the challenge, the Linux images will be restored on a Playstation 3 system, booted and examined. Note, by booting from the image, the data on disk is changed. The system log files for example will be updated and changed.

[11]From `http://www.epochconverter.com/`: The Unix epoch (or Unix time or POSIX time or Unix time stamp) is the number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds.

[12]On a UNIX system users are identified by a unique number: the UID.

The Playstation 3 is sold to consumers with its hard disk configured to be used by the Playstation OS alone. The Playstation 3 allows a "foreign" OS to use 10 GB of its hard disk space. This hard disk space can be booted by installing a special boot loader. The Petitboot bootloader[13] can boot the 10 GB partition and has the ability to boot any other bootable partition connected to the Playstation 3. These bootable partitions connected to the Playstation include bootable CDROMs, used here to install Ubuntu 8.10 from CDROM and a USB connected disk, used for booting the `jhuisi-linux-side-fs.dd` image.

The following actions were taken to boot the Linux images on the Playstation 3.

- Replace the original hard disk from the Playstation 3 with an empty hard disk.
- Boot the Playstation 3.
- Reformat the disk by using the instructions given by the Playstation 3 and create the default 10 GB partition for the OtherOS[14].
- Install Petitboot on a USB-drive and boot the Playstation 3 using the instructions on the Petitboot website.
- Create an Ubuntu Install CD-ROM from the image[15] found on the psubuntu website[16].
- Put the Ubuntu Install CD-ROM in the Playstation 3 and boot the Playstation 3.
- Choose the CD-ROM, then kernel to boot in the Petitboot menu.
- Install Ubuntu 8.10 using to the instructions found on the psubuntu website.
- After partitioning, writing the partition Table and the creation of the ext3 file system, the installation can be aborted.
- Boot the Playstation 3 and start a Petitboot-shell (press ALT-F1) from the Petitboot menu.
- Copy the Linux-filesystem images on an USB storage device having an ext3/ext2 filesystem.
- Connect the USB storage device from the previous step to the Playstation 3.
- Wait for the mounting process to finish and note the `/dev` entry for the USB storage device (`/dev/hdd` in this case).
- Unmount the `/dev/ps3da1`[17] partition.
- Restore the linux image to the hard disk:
  `dd if=/tmp/tmp/mnt/hdd1/nssal-linuxside-fs.dd of=/dev/ps3da1`.
- Reboot the Playstation 3 and choose the hard disk and the kernel to boot in the Petitboot menu.

The above method allows booting of a single image. By using an external USB Disk, it is possible to boot the other forensic image `jhuisi-linux-side-fs.dd`, without removing the on the Playstation 3 hard disk installed `nssal-linux-side-fs.dd` image. The other image must be written to a separate partition on an external USB Disk which is at least 10 GB in size. After connecting the USB disk and rebooting the Playstation 3, the Petitboot menu will show an additional hard disk which can be booted.

[13]Petitboot is available from: `http://ozlabs.org/~jk/projects/petitboot/`

[14]The 10 GB hard disk space in Linux is referenced by `/dev/ps3da`.

[15]ubuntu-8.10-alternate-powerpc+ps3.iso

[16]`http://psubuntu.com`

[17]The forensic file system images are Linux partitions `/dev/ps3da1` without the swap space `/dev/ps3hd5`.

3 Investigation

3.1 Accounts

To determine who has access to the machines the accounts were examined on both systems.

The `/etc/passwd` and `/etc/shadow` files and the directory entries in `/home` show that the `nssal-linux-side-fs.dd` image contains two interactive user accounts: nssal and jhuisi. The third field in the shadow file stores the date when the password was last changed for an account. The 'nssal' account password was last changed on December 4th 2008 and the 'jhuisi' account on March 11th 2009. Besides this the root account password was last changed on January 14th 2009.

The `nssal-linux-side-fs.dd` image also holds two accounts: jhuisi and goatboy, both passwords last changed on January 22nd 2009. The root account password on this image was also last changed on January 22nd 2009.

On both images no (daemon) accounts beside the root account were set to UID 0 (root).

Although the account passwords are not immediately needed, they could be useful in the course of the investigation. The possibility of finding encrypted files/volumes during this investigation is relatively high as the suspect has advanced knowledge of digital forensics. As recovering passwords can be a time consuming task it is sensible to start recovery at the start of the investigation. Three wordlists were created by using all strings in the `nssal-physicalmem.dd` image and the Linux images as follows:

```
#!/bin/sh
# usage: ./wordlist.sh <file>
# strings -a: Scan whole (all) files
# grep: Only list entries between 1 and 15 characters
# -x: Select only those matches that exactly match the whole line
# -E: Interpret PATTERN as an extended regular expression
strings -a $1 | grep  -x -E ".{1,15}" > wordlist_$1_temp;

# Find strings with 16-bit bigendian and append to temp file
strings -a --encoding=b $1 | grep -x -E ".{1,15}" >> wordlist_$1_temp;

# Remove duplicates
sort wordlist_$1_temp | uniq > wordlist_$1_temp2;
rm wordlist_$1_temp

# Use John the ripper to create variants for the passwords
john -stdout:25 -wordfile=wordlist_$1_temp2 -rules > wordlist_$1
rm wordlist_$1_temp2
```

The passwords are either stored in MD5 or SHA-512. In order to find the MD5 passwords John the Ripper[18] was used. First, all variations of the username were quickly tested as password in single mode with John the Ripper (JTR). JTR recovered the password 'nssal' of the nssal account on the `nssal-linux-side-fs.dd` image. Next, the generated wordlists were used. JTR was set to use the wordlists and generate and to try variations of a password (JTR rules option). Unfortunately the other MD5 password could not be found. JTR does not support SHA-512, therefore we wrote a simple Python script that is able to load a wordlist to recover the SHA-512 hashes. By using the Python script the password 'mac' for the 'jhuisi' account on the `nssal-linux-side-fs.dd` image and the password 'G04tB0y!' for the 'goatboy' account on `jhuisi-linux-side-fs.dd` image were recovered. Table 5 shows the account details of the `nssal-linux-side-fs.dd` image and Table 6 the account details for the `jhuisi-linux-side-fs.dd` image.

[18]available from: `http://www.openwall.com/john/`

Account	UID	Password	Password location	Home Directory	Last Change
root	0	-	-	`/root`	2009-01-14
nssal	1000	nssal	-	`/home/nssal`	2008-12-04
jhuisi	1001	mac	`nssal-capture-2.pcap` `jhuisi-capture-1.pcap` `nssal-physicalmem.dd`	`/home/jhuisi`	2009-03-11

Table 5: Accounts on `nssal-linux-side-fs.dd` image.

Account	UID	Password	Password location	Home directory	Last Change
root	0	-	-	`/root`	2009-1-22
jhuisi	1000	-	-	`/home/jhuisi`	2009-1-22
goatboy	1001	G04tB0y!	`jhuisi-linux-side-fs.dd` - Free space	`/home/goatboy`	2009-01-22

Table 6: Accounts on `jhuisi-linux-side-fs.dd` image.

3.2 Recovery of deleted files

Under digital forensic investigators it is common knowledge that files can be recovered by carving. However, removed files do not always have a distinctive file size, header or footer, for example the `.bash_history` file. Another method to recover files on Linux system by using the information stored in the journal.

Recovering deleted files on ext3 is not as easy as with most popular file systems. In order to ensure that ext3 can safely resume an unlink after a crash, it zeros out the block pointers in the inode, whereas ext2 just marks these blocks as unused in the block bitmaps and marks the inode as 'deleted' and does not touch the block pointers [8]. The metadata for a file is stored in an inode. File metadata includes the temporal data such as the last modified, last accessed, last changed, and deleted times. Metadata also includes the file size, user ID, group ID, permissions, and block addresses where the file content is stored [2]. Therefore it is no longer possible to determine where the file content is located when the block pointers in the inode are overwritten, see Figure 5.

The ext3 file system has a journal that records updates to the file system metadata before the update occurs. In case of a system crash, the OS reads the journal and will either reprocess or roll back the transactions in the journal so that recovery will be faster than examining each metadata structure, which is the old and slow way. The journal contains the full block that is being updated, not just the value being changed. Therefore a previous version of an inode may exist in the journal because another files was updated before the deletion [2].

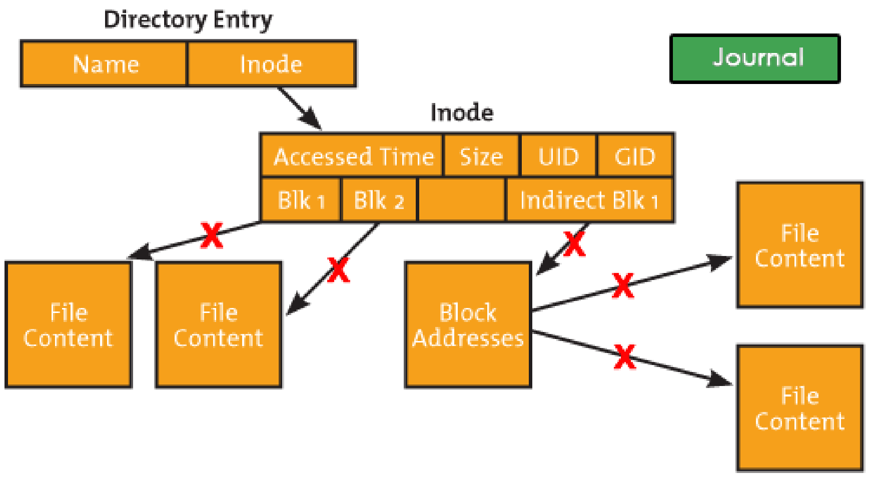


Figure 5: Relationship between the directory entry, an inode, and block of unallocated ext3 file. The links between the inode and blocks has been cleared [2]

3.2.1 The recovery

This section describes how deleted files in the home directory of the 'nssal' account on the `nssal-linux-side-fs.dd` image were recovered by using the journal, inode and directory-entry information. These methods were also used to recover other files from disk.

First, the deleted files and their inode address have to be determined. By using the utility 'debugfs' it is possible to load the image and display allocated/deleted files and their inode addressed with the command 'ls -d'. This information is loaded from the directory entry structure which is not cleared during the deletion process. Deleted files have their inode address surrounded by '<' and '>':

```
<503434> (12) 00    <503434> (12) 0     503151  (16) .themes
<SNIP>
<503148> (16) 00000    511058  (12) mod    511070  (20) .gegl-0.0
<SNIP>
 503278  (36) channels   <503148> (20) 000000000    32786  (20) crosstool
 527340  (64) d   <503434> (52) 0000000000000000    505209  (16) find.py
<SNIP>
 426276  (56) ppu-binutils-2.18.50-21.ppc.rpm   <373981> (16) ram.dd
<SNIP>
 503412  (2688) memdump-powerpc.tar   <505465> (2660) .ICEauthority-n
<505482> (20) andromachi   <505483> (2604) bateman's
<505484> (20) stanley's   <505485> (2564) stoughton's
```

By using the utility 'ext3grep' with the option '–restore-inode <inode>' ext3grep will try to find previous copies of the inode in the journal and recover the file by using the block pointer in the inode copy. If the data has not been overwritten the file can be recovered. This way inode 505482 through 505485 could be restored from the journal. The other deleted files with inode 503148, 503434 and 373981 could not be recovered this easily. 'ext3grep' also has the ability to recover files by name with '–restore-file <file>'.

In order to recover files that are not listed in the directory structure anymore (in this case the `.bash_history` file), it is possible to try to search the journal for old directory entries to determine what inode files used to have. By using 'ext3grep' and its option '–search' the command will return all blocks in the journal that match:

```
ext3grep nssal-linux-side-fs.dd --search bash_history
```

```
Running ext3grep version 0.10.1
Number of groups: 77
Minimum / maximum journal block: 1212925 / 1246235
Loading journal descriptors... sorting... done
The oldest inode block that is still in the journal, appears to be from
1236715837 = Tue Mar 10 15:10:37 2009
Number of descriptors in journal: 30544; min / max sequence numbers: 125609 / 128592
Blocks containing "bash_history": 67835 (allocated) 115671 118536....
```

Displaying all matching blocks can be quickly done by using a simple bash script that loops through all blocks [8]:

```
#!/bin/sh
blocks="67835 115671 118536 ..."

for block in $blocks; do
        ext3grep nssal-linux-side-fs.dd --ls --block $block | tee -a output.txt
done
```

The created output file shows the following information:

```
Block 338170 is a directory. The block is Allocated

          .-- File type in dir_entry (r=regular file, d=directory, l=symlink)
          |          .-- D: Deleted ; R: Reallocated
Indx Next |  Inode   | Deletion time                        Mode        File name
==========+==========+----------------data-from-inode------+-----------+=========
   0    1 d  502947                                         drwxr-xr-x  .
   1    2 d  502945                                         drwxr-xr-x  ..
   2    3 r  502948                                         rrw-r--r--  .profile
   3    4 l  502949                                         lrwxrwxrwx  Examples -> /usr/share/example-content
   4    5 r  502950                                         rrw-r--r--  .bash_logout
   5    6 r  502951                                         rrw-r--r--  .bashrc
<SNIP>
  34   35 d  503073                                         drwx------  .update-notifier
  35   36 d  505446                                         drwxr-xr-x  kmem
  36   37 r  503023                                         rrw-r--r--  .sudo_as_admin_successful
  37   38 r  503148  D 1236805424 Wed Mar 11 16:03:44 2009  rrw-------  .bash_history
<SNIP>
  46   47 r  503434  D 1236805688 Wed Mar 11 16:08:08 2009  rrw-r--r--  mem.find.pics
<SNIP>
```

The output shows that the files that were outputted by the previously used 'debugfs ls -d' command marked as deleted and named '0000', were named `.bash_history` and `mem.find.pics`. The metadata shows when the bash history was deleted and the inode number where the file was stored which is needed for the recovery attempt. The additional accessed, file modified and inode modified time stamps can be viewed with `ext3grep nssal-linux-side.dd --inode 503148`:

```
Inode is Unallocated
Group: 62
Generation Id: 440104130
uid / gid: 1000 / 1000
mode: rrw-------
size: 0
num of links: 0
sectors: 0 (--> 0 indirect blocks).

Inode Times:
Accessed:       1236793471 = Wed Mar 11 12:44:31 2009
File Modified:  1236805424 = Wed Mar 11 16:03:44 2009
Inode Modified: 1236805424 = Wed Mar 11 16:03:44 2009
Deletion time:  1236805424 = Wed Mar 11 16:03:44 2009


debugfs nssal-linux-side-fs.dd

debugfs 1.40.8 (13-Mar-2008)
```

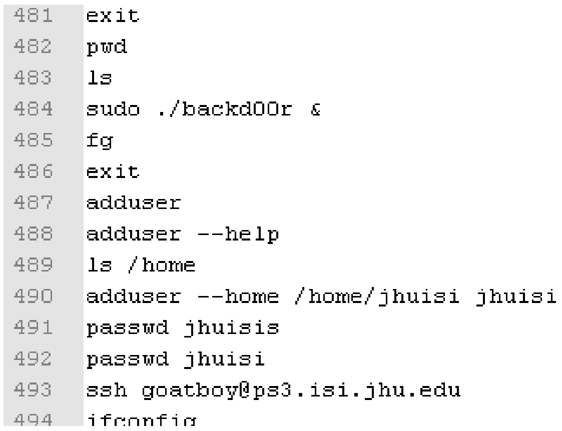
```
debugfs:  imap <503148>
Inode 503148 is part of block group 62
        located at block 2031630, offset 0x0b00

debugfs: stats
[...]
Blocks per group: 32768
[...]
```

With the group number and the block per group it is possible to export the section of the file system where the `.bash_history` file was stored. Block group 62 is needed for the recovery attempt of the bash history, therefore the block range is from `2,031,616 (62 * 32,768)` to `2,064,383 (63 * 32,768 - 1)` [2]:

```
dls nssal-linux-side-fs.dd 2031616-2064383 > unalloc.dat
```

The above 'dls' command outputs a file of 71,9 MB. By searching for common Linux commands such as 'sudo<space>' and 'ssh<space>', two continuous bash history lists were recovered. One bash history trace contains 190 commands and the other 499 commands, see Figure 6. Searching for common Linux commands was also used to extract terminal output from the `nssal-physicalmem.dd` image.



```
exit
pwd
ls
sudo ./backd00r &
fg
exit
adduser
adduser --help
ls /home
adduser --home /home/jhuisi jhuisi
passwd jhuisis
passwd jhuisi
ssh goatboy@ps3.isi.jhu.edu
ifconfig
```

Figure 6: Part of the recovered bash history

Exporting a specific block group can also be useful while trying to recover a specific file with carving to limit the number of files.

3.3 Time stamps and mount options

The Linux Ubuntu version 8.10, installed on both PlayStation 3 systems uses the mount option 'relatime', this results in a different behaviour from the default: the access time when reading a file is only updated when the original access time was less or equal to the modify or change time. Apart from the 'touch' command this setting has an impact on the forensic usage of the access time. There is no way to determine when a file or directory was read other than the first time after creation, copy, move or modification. When an USB device is auto-mounted on insertion, the mount options do not contain 'relatime' and therefore access-times are always updated. Other options to consider when a file systems has been mounted on a Linux system are 'nodiratime' and 'noatime'. 'nodiratime' does not update the access time of directories, 'noatime' does not update any access time on the mounted file system.

3.3.1 Checking mount options

For forensic use of the access, modify and change time stamps, the mount options used to mount the file system are of great importance. When a copy of the original root file system is available, the mount options in the file `/etc/fstab` can be examined to determine the mount options used to mount the file system at boot time. Note that it is always possible for the root user to change the mount options on any of the mounted file systems on any given moment by issuing the command:

mount -o remount,<options> <mountpoint>

If `/etc/fstab` is not available or not trusted, other files can be examined to determine the mount options. Files which do not change very often (or never) but are read regularly, can indicate the 'noatime' option when access time equals modify time equals change time. When these files have an access time close (seconds or minutes) to the modify time or change time the 'relatime' option is indicated. If the access time much later (days or weeks) 'atime' is indicated.

An example of a file that does not change very often but is read regularly is /etc/bash.bashrc on Ubuntu 8.10. The file /etc/bash.bashrc is read at every login but not changed very often, like wise the file `.bashrc` in the home directory of the user can be used.

Below is an example of the 'stat' of `/etc/bash.bashrc` on two Ubuntu 8.10 systems, one with the 'relatime' option and with the mount option 'atime' enabled for two days and after that 'relatime' enabled again.

relatime:

```
Access: 2009-06-04 03:06:52.000000000 -0500
Modify: 2009-03-02 09:22:30.000000000 -0500
Change: 2009-06-04 03:05:43.000000000 -0500
```

atime enabled for a 2 days:

```
Access: 2009-06-24 07:01:47.000000000 -0500
Modify: 2009-03-02 08:22:56.000000000 -0600
Change: 2009-06-04 04:40:13.000000000 -0500
```

Note the modify time, it is almost the same in both systems. This is the original creation date of the file when packaged for Ubuntu distribution. Because one system was installed using the Ubuntu server distribution and the other the Ubuntu desktop distribution the creation time do differ a little. The change time is also close to each other, both systems were installed (files copied upon the file system) about the same time. The big or little difference of the access time compared to the change time can indicate the mount options.

3.3.2 Experimentally determining mount options

What if none of the methods described in the previous section are available to determine the mount options used to mount a file system. Can the mount options be determined using 'stat' information from a (big) set of files and directories from the filesystem. The focus of this method has to be on the access time and how it differs from modify and change time. It was the indicator in the methods described in the previous section. A Python script has been created to check the access, modify and change time stamps of every file and directory in a filesystem. The output of this self-written Python script will list the of number of files where access time differs or equals from modify and change time. The script keeps track of the seconds the access time differs when access time is larger (later in time) than modify or change time. An average is calculated and shown in the output. When the script was run on the read-only mounted file system of the challenge file system dumps, the following values were retrieved.

`nssal-linux-side-fs.dd`:

```
Files: 312835
Atime < Ctime for Files: 204168  65.3%
Atime < Mtime for Files: 7354     2.3%
Atime > Ctime for Files: 54455   17.4%
Atime > Mtime for Files: 256228  81.9%
Atime = Ctime for Files: 54212   17.3%
Atime = Mtime for Files: 49253   15.7%
Average Atime diff Mtime when A>M 55048095 sec.
Average Atime diff Ctime when A>C 31838 sec.
Directories: 23792
Atime < Ctime for Directories: 11349  47.7%
Atime < Mtime for Directories: 6914   29.1%
Atime > Ctime for Directories: 10211  42.9%
Atime > Mtime for Directories: 14646  61.6%
Atime = Ctime for Directories: 2232    9.4%
Atime = Mtime for Directories: 2232    9.4%
Average Atime diff Mtime when A>M 18192838 sec.
Average Atime diff Ctime when A>C 33942 sec.
```

`jhuisi-linux-side-fs.dd`:

```
Files : 113268
Atime < Ctime for Files: 71228  62.9%
Atime < Mtime for Files: 10577   9.3%
Atime > Ctime for Files: 22646  20.0%
Atime > Mtime for Files: 85706  75.7%
Atime = Ctime for Files: 19394  17.1%
Atime = Mtime for Files: 16985  15.0%
Average Atime diff Mtime when A>M 11173094 sec.
Average Atime diff Ctime when A>C 9004 sec.
Directories: 12543
Atime < Ctime for Directories: 10522  83.9%
Atime < Mtime for Directories: 10383  82.8%
Atime > Ctime for Directories: 1818   14.5%
Atime > Mtime for Directories: 1923   15.3%
Atime = Ctime for Directories: 203     1.6%
Atime = Mtime for Directories: 237     1.9%
Average Atime diff Mtime when A>M 1640701 sec.
Average Atime diff Ctime when A>C 5418 sec.
```

Using the same Python script, the following values were retrieved from a NetBSD file system (defaults to atime) and an Ubuntu 9.04 server **without** the access time mount options: 'noatime' or 'relatime', which in effect is the same as the 'atime' of the NetBSD system. Because the NetBSD system was mounted read-write instead of read-only all directories have their access time adjusted when de files are read using 'lstat'. 'lstat' does not change any time stamps on the files it reads.

FreeBSD /usr file system (atime)

```
Files: 683486
Atime < Ctime for Files: 362776  53.1%
Atime < Mtime for Files: 505      0.7%
Atime > Ctime for Files: 251656  36.8%
Atime > Mtime for Files: 253026  37.0%
Atime = Ctime for Files: 69054   10.1%
Atime = Mtime for Files: 429955  62.9%
Average Atime diff Mtime when A>M 20037774 sec.
Average Atime diff Ctime when A>C 799615 sec.
Directories: 74143
Atime < Ctime for Directories: 0  0%
Atime < Mtime for Directories: 0  0%
Atime > Ctime for Directories: 74143  100%
Atime > Mtime for Directories: 74143  100%
Atime = Ctime for Directories: 0  0%
Atime = Mtime for Directories: 0  0%
Average Atime diff Mtime when A>M 25445529 sec.
Average Atime diff Ctime when A>C 7457875 sec.
```

Ubuntu 9.04 Server file system (atime)

```
Files: 26728
Atime < Ctime for Files: 11411  42.7%
Atime < Mtime for Files: 1758    6.6%
Atime > Ctime for Files: 10036  37.5%
Atime > Mtime for Files: 22085  82.6%
Atime = Ctime for Files: 5281   19.8%
Atime = Mtime for Files: 2885   10.8%
Average Atime diff Mtime when A>M 14020023 sec.
Average Atime diff Ctime when A>C 13245 sec.
Directories: 2697
Atime < Ctime for Directories: 1830  67.9%
Atime < Mtime for Directories: 1724  63.9%
Atime > Ctime for Directories: 720   26.7%
Atime > Mtime for Directories: 788   29.2%
Atime = Ctime for Directories: 147    5.5%
Atime = Mtime for Directories: 185    6.9%
Average Atime diff Mtime when A>M 500856 sec.
Average Atime diff Ctime when A>C 11456 sec.
```

Using the above data of the different file systems the graphs presented in Figure 7 and 8 were created. Because data of the directories in the NetBSD file system were influenced by the tool to gather the information, no directory data is not examined.

A test on two systems using 'relatime' en two using the 'atime' mount option is limited, but some remarks can be made. Further testing is needed to confirm these. In Figure 7 the percentage of files where access time is later (as in time) then change time is indicating the difference between 'relatime' and 'atime' mount options. An other indication present in this data set is the percentage of the files where access time is equal to modification time and access time equals change time. If these to values are close so access time=modify time $\approx$ access time = change time, the 'relatime' mount option was active.

Figure 8 show no direct relation between the 'noatime' and 'atime' mount options at first sight, but the difference between the value of both the average access time compared to modify

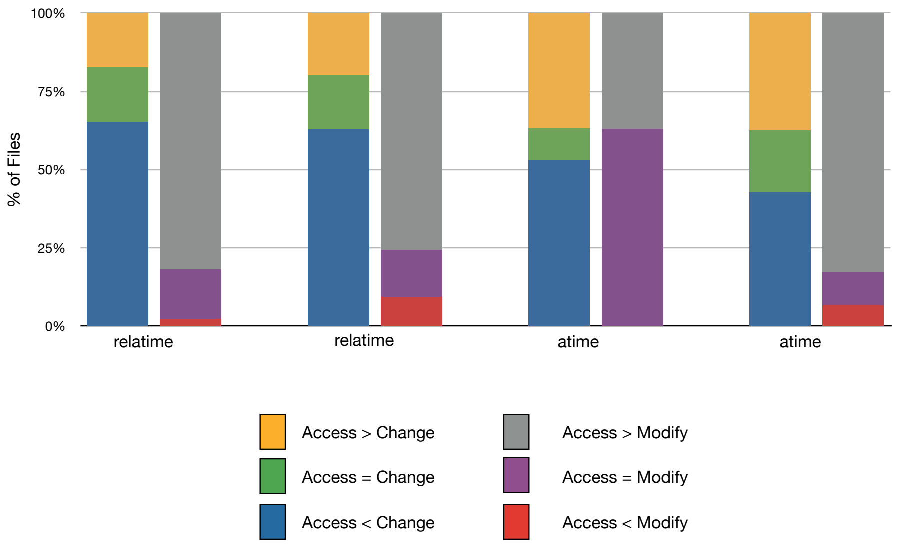


Figure 7: Access time compared with Modify and Change time if various file systems

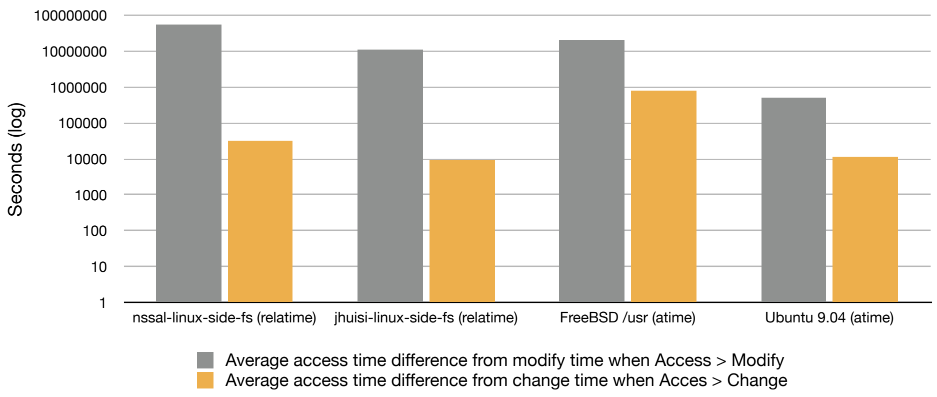


Figure 8: Average Access time difference in seconds of Modify and Change time if various file systems

and change time could be an indication.

3.4 Mardi Gras pictures

3.4.1 location and description

Mardi Gras images were found on both the file system images. On the `nssal-linux-side-fs.dd` image the pictures are located in `/home/nssal/Images`, on the `jhuisi-linux-side-fs.dd` image in `/home/jhuisi/Pictures`. Bitwise comparison of the pictures on both images showed that the pictures are equal. See Table 7 for the details of the pictures. By using the 'stat' command the access, modify and change time stamps are shown in Table 8 for the `nssal-linux-side-fs.dd` image and Table 9 for the `jhuisi-linux-side-fs.dd` image. Both the 'nssal' account on the `nssal-linux-side-fs.dd` image and the 'jhuisi' account on the `jhuisi-linux-side-fs.dd` image contain an entry in `.recently-used.xbel` for each picture. The `.recently-used.xbel` file contains a list of recently opened files in GTK[19]. The added, modified and visited time stamp associated with each picture entry in `.recently-used.xbel` match the 'access' time stamp shown by the 'stat' command.

All pictures contain the metadata fields 'creator = nssal', 'date = `2009-03-02T14:34:47-06:00`' and 'comment = Don't steal my pictures I kill you'. The date field shows a time stamp formatted in UTC -6 which matches the time zone settings of the `nssal-linux-side-fs.dd` image. This time stamp cannot match the time zone settings on the `jhuisi-linux-side-fs.dd` image as this is either UTC -4 or UTC -5. Besides this the meta data indicates that Adobe Photoshop 3.0 was used. To determine if other images were present with the meta data set, we searched for '<dc:creator> <rdf:Seq> <rdf:li>'. These fields were used to define the creator in the picture. No other images were recovered.

Filename	MD5	Size	Dimensions
3316820191_4737c3edf4.jpg	2be903dfd1b0227473bb44e05e6b77d4	133 KB	500 x 332
3318492402_731ae5cdc3_b.jpg	c670679c2b23ec2e9c31a978192b2441	238 KB	1024 x 680
3323673964_94e64ebddd_b.jpg	ea552f0c4de935f5c601bd08e91fd006	246 KB	1024 x 680
3318589824_35fe706451_b.jpg	afc59f3660d8c3bef513acc543b9efc0	262 KB	1024 x 739
3322040743_08a3b99acd_b.jpg	abfc32a61dc5893122d8801bdd983914	270 KB	1024 x 680
3320345810_6acc8185b2_b.jpg	9c7a979c001410d3291517fc5ae3b154	354 KB	1024 x 680
3321856153_0a2c9577bd_b.jpg	9ceb6ac49b474056861d78096d62859a	365 KB	1024 x 680
3317048368_639213e24b_b.jpg	77ea18321ee376c9e573386385d31634	370 KB	1024 x 680
3317820492_cefb7ca452_b.jpg	2fc920fe33403e790e9cf351ba2cbdfc	436 KB	1024 x 680
3322064459_d61e14dea5_b.jpg	1c5fe00984e2c5611da3ade638c0e5b9	462 KB	1024 x 680
3323556994_59a3982d61_b.jpg	8b4f9a823bebd52a7c48f398573f3749	730 KB	1024 x 680

Table 7: Mardi Gras pictures details that were found on both Linux images

Filename	Access	Modify	Change
3316820191_4737c3edf4.jpg	2009-3-6 15:21:32	2009-3-2 8:34:50	2009-3-6 15:21:31
3318492402_731ae5cdc3_b.jpg	2009-3-6 15:21:32	2009-3-2 8:34:50	2009-3-6 15:21:31
3323673964_94e64ebddd_b.jpg	2009-3-6 15:21:31	2009-3-2 8:34:48	2009-3-6 15:21:30
3318589824_35fe706451_b.jpg	2009-3-8 17:59:14	2009-3-2 8:34:50	2009-3-6 15:21:31
3322040743_08a3b99acd_b.jpg	2009-3-6 15:21:31	2009-3-2 8:34:50	2009-3-6 15:21:30
3320345810_6acc8185b2_b.jpg	2009-3-6 15:21:31	2009-3-2 8:34:50	2009-3-6 15:21:30
3321856153_0a2c9577bd_b.jpg	2009-3-6 15:21:31	2009-3-2 8:34:50	2009-3-6 15:21:30
3317048368_639213e24b_b.jpg	2009-3-6 15:21:32	2009-3-2 8:34:50	2009-3-6 15:21:31
3317820492_cefb7ca452_b.jpg	2009-3-6 15:21:32	2009-3-2 8:34:50	2009-3-6 15:21:31
3322064459_d61e14dea5_b.jpg	2009-3-6 15:21:32	2009-3-2 8:34:50	2009-3-6 15:21:30
3323556994_59a3982d61_b.jpg	2009-3-8 17:59:18	2009-3-2 8:34:50	2009-3-6 15:21:30

Table 8: `nssal-linux-side-fs.dd` image: Access, Modify and Change time stamps stored of the Mardi Gras pictures.

[19]GTK is a library for graphical user interfaces. GTK is used by the graphical user environment GNOME which is present on the `nssal-linux-side-fs.dd` image and the `jhuisi-linux-side-fs.dd` image.

Filename	Access	Modify	Change
3316820191_4737c3edf4.jpg	2009-3-11 11:52:17	2009-3-11 11:52:00	2009-3-11 11:52:00
3318492402_731ae5cdc3_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:01	2009-3-11 11:52:01
3323673964_94e64ebddd_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:04	2009-3-11 11:52:04
3318589824_35fe706451_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:01	2009-3-11 11:52:01
3322040743_08a3b99acd_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:02	2009-3-11 11:52:02
3320345810_6acc8185b2_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:01	2009-3-11 11:52:01
3321856153_0a2c9577bd_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:02	2009-3-11 11:52:02
3317048368_639213e24b_b.jpg	2009-3-11 11:52:17	2009-3-11 11:52:00	2009-3-11 11:52:00
3317820492_cefb7ca452_b.jpg	2009-3-11 11:52:17	2009-3-11 11:52:00	2009-3-11 11:52:00
3322064459_d61e14dea5_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:03	2009-3-11 11:52:03
3323556994_59a3982d61_b.jpg	2009-3-11 11:52:18	2009-3-11 11:52:04	2009-3-11 11:52:04

Table 9: `jhuisi-linux-side-fs.dd` image: Access, Modify and Change time stamps of the Mardi Gras pictures.

In order to determine if more Mardi Gras related images were stored on the images, WinHex was used to recover all JPEG, PNG, GIF, TIFF, Bitmap and Adobe Photoshop images. Scrolling through the images did not result in finding/recovering any other Mardi Gras related images.

3.4.2 Related traces

Although no indication of the use of steganography was found, stegdetect 0.6 was used to analyze the Mardi Gras pictures for steganographic content to rule out that hidden information was exchanged or stored in the pictures. It runs statistical tests to determine if steganographic content is present, and also tries to find the system that has been used to embed the hidden information. Stegdetect detected two false positives in picture `3316820191_4737c3edf4.jpg` and `3323556994_59a3982d61_b.jpg`. Stegbreak was used with the created wordlists without any results.

The filenames of the Mardi Gras pictures were found as deleted file entries on the thumb drive. Besides this, the recovered bash history of the 'nssal' account and matching backdoor[20] commands found in `nssal-capture-1.pcap` and `jhuisi-capture-1.pcap` (see section 3.9.8) shows a 'cd' command into the Images directory of the 'nssal' account, but no other commands. Furthermore, `nssal-capture-1.pcap` shows network traffic to Mardi Gras related websites, see section 3.9.

Keyword searches such as '/Images', '/Pictures' and 'mardi gras' did not reveal any other information. No related deleted files were recovered.

3.5 Drug recipes

3.5.1 Location and description

Drug recipes were found on both the `nssal-linux-side-fs.dd` and the `jhuisi-linux-side-fs.dd` image. On the `nssal-linux-side-fs.dd` image the recipes are located in `/home/nssal/Recipes`, on the `jhuisi-linux-side-fs.dd` image in `/home/goatboy/Recipes`. The `jhuisi-linux-side-fs.dd` image contains one recipe more than the `nssal-linux` image named shéyòu. Bitwise comparison of the other four drug recipes on both images showed that the recipes are equal. See Table 10 for the details of the drug recipes. Table 11 lists drug recipes accessed, modified and changed time stamps for `nssal-linux-side-fs.dd`, Table 12 shows the `jhuisi-linux-side-fs.dd` time stamps.

The 'Recipes' directory on the `jhuisi-linux-side-fs.dd` image also contained a sub-directory named 'customers'. The 'customer' directory holds a file named 'Tír na nÓg' which contains a

[20]Software used to gain unauthorized access to a system.

Filename	MD5	Size
andromachi	`3f262643d1f6d50a77e860c9a8434e28`	1,7 KB
bateman’s	`bc517b62873a8abf9349e4c86f0bb11c`	140 B
shéyòu	`940734945d3c16b0b79660c7c0aaeeb1`	215 B
stanley’s	`36c058ced5f379a9e5042da2e6ca2bf9`	235 B
stoughton’s	`b5d615dba78bbc7e0cdcf29ebe82962d`	359 B

Table 10: Drug recipe details that were found on both Linux images.

Filename	Accessed	Modified	Change
andromachi	`2009-03-11 11:49:40`	`2009-03-11 11:49:40`	`2009-03-11 11:53:24`
bateman’s	`2009-03-11 11:49:49`	`2009-03-11 11:49:49`	`2009-03-11 11:53:29`
stanley’s	`2009-03-11 11:50:05`	`2009-03-11 11:50:05`	`2009-03-11 11:52:52`
stoughton’s	`2009-03-11 11:50:10`	`2009-03-11 11:50:10`	`2009-03-11 11:52:57`

Table 11: Time stamps of drug recipes found in `/home/nssal/Recipes` on `nssal-linux` image.

Filename	Accessed	Modified	Change
andromachi	`2009-03-11 11:49:38`	`2009-03-05 15:05:41`	`2009-03-06 09:15:55`
bateman’s	`2009-03-11 11:45:42`	`2009-03-05 15:01:15`	`2009-03-06 09:15:55`
shéyòu	`2009-03-11 11:51:13`	`2009-03-06 09:16:10`	`2009-03-06 09:15:55`
stanley’s	`2009-03-11 11:50:02`	`2009-03-05 14:49:06`	`2009-03-06 09:15:55`
stoughton’s	`2009-03-11 11:50:08`	`2009-03-05 14:45:03`	`2009-03-06 09:15:55`

Table 12: Time stamps of drug recipes found in `/home/goatboy/Recipes` on `jhuisi-linux` image.

list of names and addresses. Besides the ’Tír na nÓg’ on the `jhuisi-linux-side-fs.dd` image, the names and addresses were not found elsewhere.

3.5.2 Related traces

The `/home/goatboy/Recipes` directory on the `jhuisi-linux-side-fs.dd` contains a deleted directory entry for the file ’recipes.tar’; this file could not be recovered. No traces of ’recipes.tar’ were found on the other images.

The `/home/goatboy/.bash_history` on `jhuisi-linux-side-fs.dd` shows traces of unpacking of ’recipes.tar’ and the creation ’customer’ directory. Besides this, it shows interaction between the ’goatboy’ and ’jhuisi’ accounts which appears to be related to the drug recipes. The `.bash_history` file of the ’goatboy’ account is listed below:

```
1 ls
2 mkdir Recipes
3 ls
4 cd Recipes/
5 ls
6 tar xvf recipes.tar
7 ls customers/
8 mv Tir\ na\ nOg customers/
9 ls
10 rm recipes.tar
11 exit
12 ls
13 ls -l Recipes/
14 exit
15 ls
16 cd Examples
17 ls
18 cd ..
19 clear
20 ls
21 cd Recipes
22 ls
23 cat bateman\'s
24 talk
25 ytalk
26 ntalk
27 write jhuisi
28 ls
29 cat custmers
30 write jhuisi
31 cat customers
32 cd customers
33 ls
34 clear
35 ls -l
36 cat Tir\ na\ nOg
37 ls
38 write jhuisi
39 I feel younger already!
40 write jhuisi
41 write
42 write jhuisi
43 Nice doing business with you!
44 clear
45 ls
46 cd
47 clear
48 ls
```

The first six entries of the bash history given above shows that user 'goatboy' created the directory 'Recipes', changed into this directory and subsequently unpacked 'recipes.tar'. Apparently 'recipes.tar' is created by another user as a directory is empty when it is created. However, the bash history of the 'jhuisi' is not in human readable format for a large part and does not show such activity. The bash history of the 'root' account also does not contain any drug recipe related entries.

Several terminal traces were found in the `nssal-physicial-mem.dd` image. Terminal traces were found by searching for '@ps3:','@nssal-ps3:' and '<user>@'. These traces indicate that an SSH session was set-up from the `nssal-linux-side-fs.dd` image to the `jhuisi-linux-side-fs.dd` image with the user 'goatboy'. Terminal traces from the `nssal-physicial-mem.dd` image:

```
Terminal - goatboy@ps3: ~/Recipes/customers

goatboy@ps3:~$ ls
Examples  Recipes
goatboy@ps3:~$ logout
Connection to ps3.isi.jhu.edu closed.


goatboy@ps3:~/Recipes/customers$ clear
goatboy@ps3:~/Recipes/customers$ ls
goatboy@ps3:~/Recipes/customers$
goatboy@ps3:~/Recipes/customers$ write jhuisi
write: jhuisi is logged in more than once; writing to pts/3
Slurp!
Slurp!
goatboy@ps3:~/Recipes/customers$
```

The write commands found in the bash history of the user 'goatboy' and the terminal traces in `nssal-physicial-mem.dd` suggest that some kind of exchange took place related to the drug recipes.

3.6 Backdoor

3.7 Location and description

On the `jhuisi-linux-side-fs.dd` image the files `backd00r.c`, `.backd00r.c.swp`, `.backd00r.c.swx` and `a.out` were found in the directory `/home/jhuisi`. The contents of `backd00r.c` shows that the file is the source code of software to gain unauthorized access to a system. By compiling the `backd00r.c` file with 'gcc' and bit comparing the output to the `a.out` file, showed that the files are equal. The files `.backd00r.c.swp`, `.backd00r.c.swx` and `a.out` show that backdoor software was compiled and edited on the `jhuisi-linux-side-fs.dd` image.

A file named `backd00r` was recovered in the directory `/home/jhuisi` on the `nssal-linux-side-fs.dd` image. No other `backd00r` files were found. Bit comparing this `backd00r` file with the `a.out` file on the `jhuisi-linux-side-fs.dd` image shows that the files are equal.

Table 13 list all time stamp of the backd00r files.

Filename	**dd**	**Accessed***	**Modified***	**Changed***	**Deletion***
backd00r	N	12:08:59	12:36:15	12:36:15	12:36:15
backd00r.c	J	11:55:16	11:55:06	11:55:06	not deleted
.backd00r.c.swp	J	13:32:53	13:52:53	13:52:53	13:52:53
.backd00r.c.swx	J	12:20:34	12:20:34	12:20:34	unavailable
a.out	J	11:58:08	11:58:07	11:58:07	unavailable

* The date is 2009-03-11

Table 13: Details of backd00r files. the 'dd' field shows on which image the files were found; value 'N' denotes `nssal-linux-side-fs.dd` image and 'J' denotes `jhuisi-linux-side-fs.dd`.

3.7.1 Related traces

The below listed bash history of the 'jhuisi' account on the `nssal-linux-side-fs.dd` image shows that the `backd00r` was executed by the 'jhuisi' user. The 'telnet', 'ps', 'netstat' commands were probably used to check if the backdoor was running properly.

```
1 ls                            10 ls                           19 netstat -an
2 cd Examples                   11 cd                           20 netstat -an | more
3 ls                            12 ls                           21 telnet localhost 45541
4 cd                            13 ./backdoor                   22 ps auxww | grep klog
5 cd ..                         14 ls -l                        23
6 ls                            15 ~/backdoor                   24 ps auxww | grep back
7 cd nssal/                     16 ls -l                        25 netstat -an | more
8 ls                            17 sudo nssal ./backd00r        26 exit
9 cd Images/                    18 su
```

The recovered bash history of the 'nssal' account also shows that the backdoor was executed by the user 'nssal':

```
sudo ./backd00r &
exit
pwd
ls
sudo ./backd00r &
fg
exit
```

The log file `/var/log/auth.log` contains the sudo attempts that are shown in the above bash history:

```
2009-03-11 12:01:35        jhuisi : user NOT in sudoers ; TTY=pts/2 ; PWD=/home/jhuisi ;
                              USER=root ; COMMAND=nssal ./backd00r
2009-03-11 12:08:59        nssal : TTY=pts/3 ; PWD=/home/jhuisi ; USER=root ; COMMAND=./backd00r
```

The first entry shows that the 'jhuisi' user is not authorized to run the backdoor with administrator privileges. Seven minutes later the user 'nssal' runs the backdoor successfully with administrator privileges. Tests with the backdoor software in a virtual machine show that administrator privileges are required to be able remotely login with root privileges. The user 'jhuisi' does not have administrator privileges.

Backdoor traffic was found in `nssal-capture-2.pcap` and `jhuisi-capture-1.pcap`, see section 3.9.8. Searching for '[backdoor]$' (as found in the network traffic) in all images did not reveal any extra given backdoor commands.

3.8 Thumb drive

The command 'blkid' shows the thumb drive's ID: '14D0-6139'. The `/var/log/auth.log` file on the `nssal` image shows that the a SanDisk 512 MB thumb drive with ID '14D0_6139' was mounted on 2009-03-06 15:19:55 for 32 minutes until 15:48:51. The thumb-drive does not have a label. No other traces of thumb ID were found on the images.

The `nssal-thumb` drive contains traces of storage of the same 11 Mardi Gras pictures that were found on the Linux images. The file entries and all relevant time stamps are listed in Table 14. The time stamps show that on March 2nd and March 6th events took place. The pictures - or pieces of the pictures - could not be extracted from the data on the thumb drive. Looking at the size of the files shows that pictures of March 6th are a few (between 6 and 10) kilobytes bigger than the files of March 2nd. The files stored on the Linux images are sized exactly between the size of the file on March 2nd and its size on the March 6th.

Filename	Size	Created	Modified	Accessed
3323673964_94e64ebddd_b.jpg	242 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3322064459_d61e14dea5_b.jpg	464 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3323556994_59a3982d61_b.jpg	0,7 MB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3321856153_0a2c9577bd_b.jpg	368 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3322040743_08a3b99acd_b.jpg	272 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3318492402_731ae5cdc3_b.jpg	240 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3318589824_35fe706451_b.jpg	264 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3320345810_6acc8185b2_b.jpg	360 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3316820191_4737c3edf4.jpg	136 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3317048368_639213e24b_b.jpg	376 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3317820492_cefb7ca452_b.jpg	440 KB	2009-03-06 21:37	2009-03-06 21:37	2009-03-06
3317820492_cefb7ca452_b.jpg	432 KB	2009-03-02 14:24	2009-03-02 14:24	2009-03-02
3317048368_639213e24b_b.jpg	366 KB	2009-03-02 14:21	2009-03-02 14:21	2009-03-02
3316820191_4737c3edf4.jpg	129 KB	2009-03-02 14:21	2009-03-02 14:21	2009-03-02
3318492402_731ae5cdc3_b.jpg	234 KB	2009-03-02 14:20	2009-03-02 14:20	2009-03-02
3318589824_35fe706451_b.jpg	258 KB	2009-03-02 14:19	2009-03-02 14:19	2009-03-02
3320345810_6acc8185b2_b.jpg	350 KB	2009-03-02 14:19	2009-03-02 14:19	2009-03-02
3321856153_0a2c9577bd_b.jpg	361 KB	2009-03-02 14:17	2009-03-02 14:17	2009-03-02
3322040743_08a3b99acd_b.jpg	266 KB	2009-03-02 14:17	2009-03-02 14:17	2009-03-02
3322064459_d61e14dea5_b.jpg	458 KB	2009-03-02 14:16	2009-03-02 14:16	2009-03-02
3323556994_59a3982d61_b.jpg	0,7 MB	2009-03-02 14:16	2009-03-02 14:16	2009-03-02
3323673964_94e64ebddd_b.jpg	242 KB	2009-03-02 14:15	2009-03-02 14:15	2009-03-02

Table 14: `nssal-thumb` images: FAT Created, Modified and Accessed timestamps of the Mardi Gras pictures dived in two events.

Besides the picture entries, the thumb drive also holds a deleted file entry '?hatever' of 488 MB. Deleted files on the FAT file system are marked as deleted by changing the first byte of the file entry to 0xE5, therefore this file was probably named 'whatever'. The 'whatever' file has '2009-03-02 14:03:15' stored as the creation time stamp and '2009-03-02 14:06:50' as the modified time stamp. Recovered bash history on the `nssal` image shows a memory dump named named 'whatever' was created and shredded. No other traces of the file 'whatever' have been found.

The time stamps of the files and the first sector were each file was stored shows that the 'whatever' file was the first file that was created on the drive. Subsequently, the 'whatever' file was presumably removed as a few minutes later the file '3323673964_94e64ebddd_b.jpg' was created in same sector (sector 528). On the sixth of March a new batch of pictures was stored on the drive starting at sector 528 indicating that the old data was removed.

Looking at the data stored in the sectors shows that sector 528-8335 holds data with a high entropy, see Figure 9. The size of the data in these sector is almost equal to the sum of all MG pictures of the sixth of March (3,81 MB). Next, blocks of 610 bytes filled with zeros (0x30) are stored until sector 999694. The blocks of 610 bytes are separated by two bytes 0x0d0a which represent a Windows line break. Selecting all data of sector 528 until the last 0x30 byte in sector 999694, sums a total of exactly 488 MB (the size of the 'whatever' file). Subsequently, sector 999695-999703 contain data with a high entropy. The last sector of the drive is empty. This is illustrated in Figure 10.

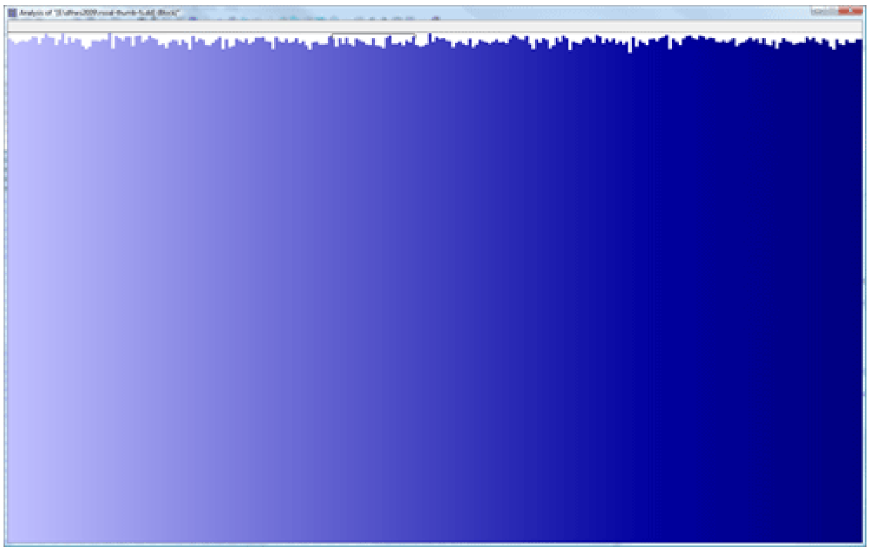


Figure 9: WinHex showing the occurences of bytes in sector 528-8335, which shows that the data is random.

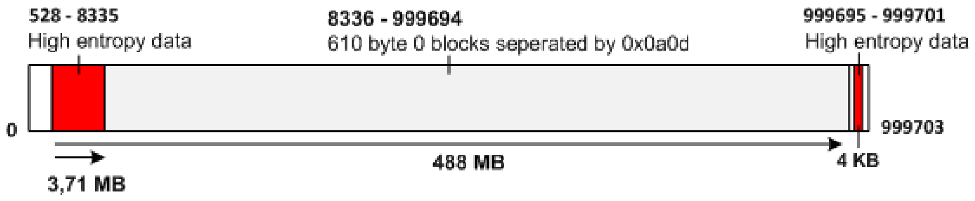
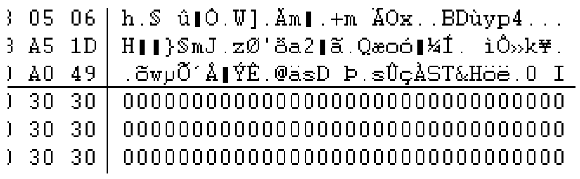


Figure 10: Data on thumb drive

The random data could be an encrypted container or a compressed archive. However, no related traces to cryptographic activities have been found. An other possibility is that the data was overwritten with random data, which looks the most plausible. The high entropy data of sectors 528-8335 is exactly written to a sector boundary, see Figure 11. Some tests with the command 'shred -u' show that random bytes are generated untill boundary offsets. The command 'shred -u' was found several times in the bash history of the 'nssal' account. However, no 'shredding' history directly related to the thumb drive has been found.

Trying to identify the data on the drive with 'file –keep-going –raw' and searching for bytes from the drive on the Linux images was fruitless. No data could be recovered from the thumb drive.



```
3 05 06 | h.S û▮Ô.W].Åm▮.+m ÄOx..BDùyp4...
3 A5 1D | H▮▮}SmJ.zØ'ða2▮ã.Qæoó▮¼Í. iÔ»k¥.
) A0 49 | .õwµÕ´Å▮ŸÊ.@äsD Þ.sÛçÀST&Höë.0 I
--------+----------------------------------
) 30 30 | 00000000000000000000000000000000
) 30 30 | 00000000000000000000000000000000
) 30 30 | 00000000000000000000000000000000
```

Figure 11: boundary between sector 8335 and 8336.

3.9 Network traffic

This section describes the investigations done on the available network traffic captures. First the time stamps within the captures and the capture periods are examined in subsection 3.9.1. Next, the MAC address and IP address combinations when Linux was booted are summarized in 3.9.2. Then the various types of traffic found within the captures are specified in subsections. Playstation 3 and Playstation Portable traffic are addressed in subsection 3.9.3 and the related Playstation Home captured traffic in subsection 3.9.4. The suspect's Playstation visited websites related to Mardi Gras, section 3.9.5 describes this. The SSH sessions found in the network traffic and the known_hosts traces left by these SSH sessions are specified in subsection 3.9.6 and 3.9.7 respectively. Finally the traffic generated by the backdoor software (section 3.6) is specified in section 3.9.8.

Due to time constrains an in depth analysis of all the network traffic was not possible.

3.9.1 Network time stamps

The network traces contain a UTC time stamp for each captured packet. These time stamps are created by the device used for capturing the data. The devices used for the capture had their own clock source and were not synchronized. Table 15 lists the capture periods that are extracted from the network data.

Network Capture	Start	End	Length
`nssal-capture-1.pcap`	`2009-03-05 14:18:52`	`2009-03-05 14:42:53`	`00:24:01`
`nssal-capture-2.pcap`	`2009-03-11 11:01:16`	`2009-03-11 12:16:13`	`01:14:57`
`jhuisi-capture-1.pcap`	`2009-03-11 11:01:44`	`2009-03-11 12:16:40`	`01:14:56`

Table 15: Time periods extracted from the network captures.

Table 15 shows that the `nssal-capture-2.pcap` and `jhuisi-capture-1.pcap` contain network traffic of the same period. All traffic found between the suspect's Playstation and JHUISI Playstation appears to be either SSH or backdoor traffic. This is covered in section 3.9.6 and section 3.9.8 respectively.

By examining `nssal-capture-2.pcap` and `jhuisi-capture-1.pcap`, SSH traffic was found between the suspect's Playstation and the JHUISI Playstation. The following parts of the captures can be used to determine the clock skew between the two systems used to capture the data:

nssal-capture-2.pcap:
35804 2009-03-11 **11:51:57.409715** 128.220.249.83 137.30.123.40 TCP 51942 > ssh [SYN]
35805 2009-03-11 **11:51:57.409862** 137.30.123.40 128.220.249.83 TCP ssh > 51942 [SYN, ACK]

jhuisi-capture-1.pcap:
166834 2009-03-11 **11:52:24.069081** 128.220.249.83 137.30.123.40 TCP 51942 > ssh [SYN]
166837 2009-03-11 **11:52:24.118586** 137.30.123.40 128.220.249.83 TCP ssh > 51942 [SYN, ACK]

Looking at the above TCP/IP conversations that are present in both captures, the time skew between the captures was determined to be around 26 seconds. Both captures started and ended with one second difference.

3.9.2 MAC and IP addresses

The IP and MAC addresses that the two Playstations used during when Linux was booted, were determined by using the `/var/log/syslog` file. The file `/var/log/daemon.log` was used to find the DHCP settings and the `/etc/network/interfaces` for the static configuration of the network devices. The host names can be found in the file `/etc/hostsname`. Table 16 gives an overview of the settings.

Image	hostname	MAC	IP

`nssal-linux-side-fs.dd`	nssal-ps3	`00:1f:a7:b2:1a:de`	`137.30.123.176 (Mar 5)`
			`137.30.123.40 (Mar 11)`

`jhuisi-linux-side-fs.dd`	ps3	`00:1f:a7:5f:34:a0`	`128.220.249.83`

Table 16: Host name, MAC- and IP-address of the Playstation 3 systems when Linux was booted.

3.9.3 Playstation and PSP Traffic

Network traffic generated by either Playstation 3 (PS3) or Playstation Portable (PSP) operating system were found in all three network captures.

All network traffic in the capture `nssal-capture-1.pcap` of March 5th (Table 15) was generated by two IP addresses: 137.30.123.78 and 137.30.123.170. The IP address 137.30.123.78 is the suspect's Playstation 3 running the Playstation OS. Log files on `nssal-linux-side-fs.dd` show that the combination of the MAC address `00:1f:a7:b2:1a:de` and the IP address 137.30.123.78 was not used when Linux was booted, as seen in Table 16. This implicates the suspect's Playstation was running the Playstation OS with IP address 137.30.123.78. The presence of `User-Agent: Mozilla/5.0 (PLAYSTATION 3; 1.00)` in the HTTP(S) traffic can confirm this. The IP addresses that the suspect's system connects to are all related to Playstation websites, a Playstation 3 update site or websites related to Madri Gras. The Mardi Gras related traffic is specified in section 3.9.5). Most of the Playstation websites visited are Playstation Home related, detailed in section 3.9.4. The IP address 137.30.123.170 connects to either Playstation websites or the Playstation Portable update site. Connection to the Playstation Portable update site is unique for PSP generated traffic and therefore it is believed the device with IP 137.30.123.170 is a PSP.

The network traffic on the 11th of March of both Playstation systems contain Playstation Home traffic (see section 3.9.4), SSH traffic (see section 3.9.6) and backdoor traffic (see section 3.9.8). The Playstation Home traffic occurred the first 25 minutes when both the NSSAL and JHUISI system where running the Playstaton OS. Around 11:30 both systems stopped generating Playstation Home traffic and booted into Linux from where SSH and backdoor traffic was generated. Apart from SSH traffic between the NSSAL system and the JHUISI system, a third system (IP: 128.220.251.228) connected to the JHUISI system using SSH, see section 3.9.6.

3.9.4 Playstation home

PlayStation Home is a 3D social gaming community that allows Playstation 3 users to meet, chat, plan, and launch into games together. The Playstation home environment is accessible on any playstation 3 under the Playstation network menu. Playstation Home is not available for the Playstation Portable [5]. Logging in on Playstation home and capturing and examining the network traffic, shows that the captured traffic that is found in all three network captures and that both the suspect and the person on the JHUISI Playstation have used Playstation Home.

Capturing Playstation Home log-in session, shows that the account that is used to log-in appears several times in the format 'AcctName=<account_name>'. The used account names were found by searching all packet bytes in all captures for the string 'AcctName='. The `nssal-capture-1` and `nssal-capture-2` contain login traces of the user 'nssal', the `jhuisi-capture-1` holds traces of the account 'jhuisi'.

Walking around in the virtual Playstation Home world, shows that it is possible to load profiles of other users that are near by your character. The profile contains information such as the avatar, spoken languages, 'about me' information and trophies. The network traffic shows that a XML file is downloaded from the Playstation server when a profile is viewed. It is also possible to view your own profile in this case a XML file is downloaded in the same format.

Table 17 lists the account names to log-in and the profiles that were viewed in each capture.

Capture	Viewed profiles	Used account name
`nssal-capture-1`	jhuisi narutowy79	nssal
`nssal-capture-2`	narutowy79 jhuisi nssal	nssal
`jhuisi-capture-1`	jhuisi	jhuisi

Table 17: Playstation Home viewed profiles

No other interesting information besides the username was found in the profiles. Profile information of user 'narutowy79' was found several times in `nssal-capture-1.pcap` and `nssal-capture-2.pcap`. Searching for 'narutowy79' on all evidence images returned no results.

Playstation Home chat traffic could not be recovered as it is not send in human readable format.

3.9.5 Mardigrass related traffic

Capture `nssal-capture-1.pcap` created on March 5th contains traffic related to the Mardi Gras websites. Websites like 'www.holidays.net/mardigras', 'mardigras.makeparties.com', 'www.huffingtonpost.com' and 'www.newsday.com' where visited. These sites served content related to Mardi Gras and pictures of the resent Mardi Gras festivities of 24 February 2009. All sites are publicly available. No traces were found that interaction with between the suspect and other people took place through these websites.

3.9.6 SSH network traffic

Wireshark was used to view SSH traffic between end points (called conversations in Wireshark) and summarize the packets sent in each direction. No SSH network traces were found in the `nssal-capture-1.pcap` capture. Table 18 shows SSH traffic in `nssal-capture-2.pcap` capture and Table 19 the SSH traffic in `jhuisi-capture-1.pcap`. The first remark in the 'remarks' field holds the SSH user which has been used to initiate the session. The user was determined by using the logs on the file system images. The remark 'possible SCP' is present if an SSH session was successful and lasted shortly.

nr.	Remarks	Start Time	End Time	Src IP:Port	Dst IP:22	Bytes A− >B	Bytes A< −B
1	Failed goatboy	`2009-03-11 11:42:15`	`11:44:26`	`137.30.123.40:35892`	`128.220.249.83`	`3019`	`3323`
2	SSH goatboy	`2009-03-11 11:44:29`	`12:14:15`	`137.30.123.40:35893`	`128.220.249.83`	`178011`	`137353`
3	SFTP goatboy	`2009-03-11 11:49:02`	`11:50:09`	`137.30.123.40:34612`	`128.220.249.83`	`8831`	`11801`
4	SSH jhuisi	`2009-03-11 11:49:37`	`12:09:45`	`128.220.249.83:51874`	`137.30.123.40`	`72037`	`119641`
5	- SSH juisi - Possible SCP - Only session that fits all MG pics.	`2009-03-11 11:51:57`	`11:52:04`	`128.220.249.83:51942`	`137.30.123.40`	`109937`	`4167015`
6	SSH jhuisi Possible SCP	`2009-03-11 11:56:06`	`11:56:09`	`128.220.249.83:51075`	`137.30.123.40`	`22685`	`4623`
7	SSH nssal	`2009-03-11 12:02:23`	`12:11:00`	`128.220.249.83:51109`	`137.30.123.40`	`24495`	`23281`

Table 18: SSH sessions within the `nssal-capture-2.pcap` (no skew correction).

nr.	Remarks	Start Time	End Time	Src IP:Port	Dst IP:22	Bytes A− >B	Bytes A< −B
1	Not between N and J	`2009-03-11 11:31:43`	`11:32:04`	`128.220.251.228:9465`	`128.220.249.83`	`6935`	`4227`
2	Not between N and J	`2009-03-11 11:32:18`	`11:33:08`	`128.220.251.228:9477`	`128.220.249.83`	`8085`	`8599`
3	Failed goatboy	`2009-03-11 11:42:42`	`11:44:53`	`137.30.123.40:35892`	`128.220.249.83`	`2953`	`3257`
4	SSH goatboy	`2009-03-11 11:44:56`	`12:14:42`	`137.30.123.40:35893`	`128.220.249.83`	`178011`	`134669`
5	Not between N and J	`2009-03-11 11:45:09`	`11:45:42`	`128.220.251.228:10250`	`128.220.249.83`	`4965`	`5353`
6	SFTP goatboy	`2009-03-11 11:49:28`	`11:50:36`	`137.30.123.40:34612`	`128.220.249.83`	`8831`	`11801`
7	SSH jhuisi	`2009-03-11 11:50:04`	`12:10:11`	`128.220.249.83:51874`	`137.30.123.40`	`66189`	`119641`
8	- SSH jhuisi - Possible SCP - Only session that fits all MG pics.	`2009-03-11 11:52:24`	`11:52:30`	`128.220.249.83:51942`	`137.30.123.40`	`109481`	`4165581`
9	SSH jhuisi Possible SCP	`2009-03-11 11:56:32`	`11:56:35`	`128.220.249.83:51075`	`137.30.123.40`	`22685`	`4632`
10	SSH nssal	`2009-03-11 12:02:49`	`12:11:27`	`128.220.249.83:51109`	`137.30.123.40`	`23109`	`23281`

Table 19: SSH sessions within the `jhuisi-capture-1.pcap` (no skew correction).

The rows with the yellow backgrounds in the above Tables are interesting. This session only lasted a few seconds and many bytes were sent which characterizes a SCP session. A total of approximately 3,97 MB (4167015 bytes) were sent from one direction to the other. The total size of all Mardi Gras pictures that were encountered on the file systems is 3,8 MB, see section 3.4. To test how many bytes are actually transmitted, Mardi Gras pictures were copied with SCP from one computer to another. Three tests showed that approximately 3,95 MB are transmitted. Therefore, this session could be a SCP session in which all the Mardi Gras pictures were copied. Looking at the total transmitted bytes of the other sessions, the yellow session is the only one that could have fitted all the Mardi Gras pictures.

To check if deciphering the SSH sessions was possible, the host keys used for the SSH sessions were tested using the Linux utility ssh-vulnkey[21] without result. Running the utilities AESKeyFinder and RSAKeyFinder found on the website[22] of the writers of the paper *Lest we remember: cold-boot attacks on encryption keys*[3] on the `nssal-physicalmem.dd` did not recover any AES or RSA keys.

3.9.7 known_hosts files

In `nssal-linux-side-fs.dd` the `/root/.ssh/known_hosts` and `/home/nssal/.ssh/known_hosts` are available. In the `jhuisi-linux-side-fs.dd` the file `/home/jhuisi/.ssh/known_hosts` was found, no other traces of files named known_hosts were found. An overview of the time stamps of the know_hosts files and the hosts keys they contain is given in Table 20.

Image	location	Access*	Modify*	Change*	Has Key of host
`nssal-linux-side-fs.dd`	`/root`	11:44:32	11:42:21	11:42:21	JHUISI
`nssal-linux-side-fs.dd`	`/home/nssal`	11:49:18	11:49:18	11:49:18	JHUISI
`jhuisi-linux-side-fs.dd`	`/home/jhuisi`	11:51:57	11:49:40	11:49:40	NSSAL

* The date is 2009-03-11

Table 20: Time stamps of and host keys in .ssh/known_hosts

3.9.8 backdoor traffic

As mentioned in section 3.6, a file `backd00r.c` was found on the `jhuisi-linuxside-fs.dd` image. This file contains the code:

# define PORT 45541

This is the TCP/IP Port used by the backd00r program to allow access from a remote location. Both the `nssal-capture-2.pcap` shown in Table 21 and `jhuisi-capture-1.pcap` in Table 22, show TCP/IP traffic to this port on the suspect's Playstation initiated from the JHUISI Playstation.

Start Time	End	Src IP:Port	Dst IP:Port	Pkts/Bytes
2009-03-11 12:09:59	12:16:13	128.220.249.83:56151	137.30.123.40:45541	102/9015

Table 21: backd00r sesions within the `nssal-capture-2.cap` (no skew correction).

[21]The ssh-vulkey utility tests the hosts keys against a blacklist of keys known to be compromised and includes the host keys created by openssh with the predictable random number generator bug: `http://lists.debian.org/debian-security-announce/2008/msg00152.html`.

[22]`http://citp.princeton.edu/memory/code/`

Start Time	End	Src IP:Port	Dst IP:Port	Pkts/Bytes
2009-03-11 12:10:26	12:16:40	128.220.249.83:56151	137.30.123.40:45541	101/8941

Table 22: backd00r sesions within the `jhuisi-capture-1.cap` (no skew correction).

The backd00r traffic is "telnet"-like and can therefore be inspected using the "follow TCP stream" option in the Wireshark network inspection tool. The output below shows some commands of the backdoor session. The complete output of the commands given in the '[backd00r]' prompt is omitted in this overview to avoid lengthy output.

```
jhuisi

Backdoor by darkXside

Enter the second password.
mac

Password accepted!
[backdoor]# ls
[backdoor]# ls
[backdoor]# rm backd00r
[backdoor]# ls
[backdoor]# cd ..
[backdoor]# cd nssal
[backdoor]# ls
[backdoor]# who
[backdoor]# ls
[backdoor]# cd Recipes
[backdoor]# ls
[backdoor]# ls Videos
[backdoor]# ls screenshots
[backdoor]# ls Images
[backdoor]# fdisk -l
[backdoor]# ls -l /media
[backdoor]# ls -l /mnt
[backdoor]# ls -l /mnt/usb
[backdoor]#
```

3.10 Application history

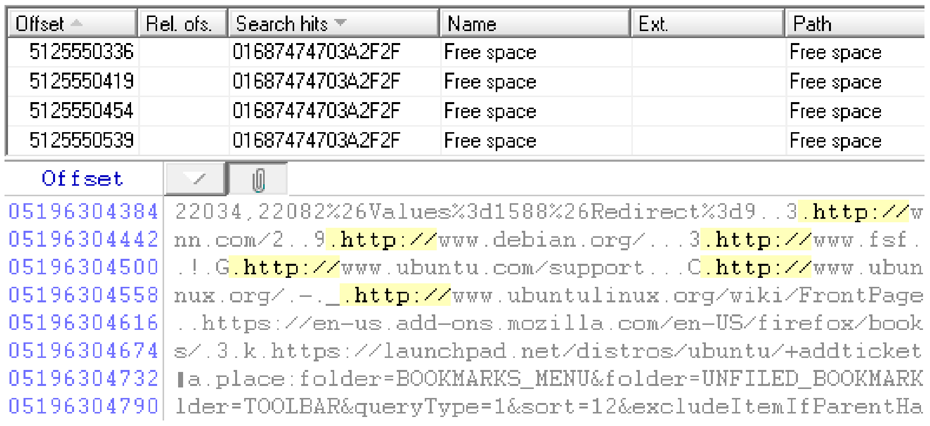
Application history is usually stored in the home directory of a user. Application history was searched for by looking in the home directories of all accounts and starting applications on the Playstation (with the Linux images restored on a hard disk). On the `nssal-linux-side-fs.dd` image two browsers were used by the 'nssal' account: Mozilla Firefox and Opera. The `jhuisi-linux-side-fs.dd` image only show traces of Firefox by the 'jhuisi' account. No traces were found for any other applications.

3.10.1 Firefox

Firefox stores its settings in `/home/-user-/.mozilla/firefox/`. FoxAnalysis[23] was used to extract Firefox's Website History, Bookmarks, Cookies, Downloads and Form History from the SQLite files. The cache was examined with MozillaCacheView[24]. Firefox traces on the `nssal-linux-side-fs.dd` image indicate that Opera was downloaded and that the suspect visited websites with regards to 64 bit compiling, the Cell processor, PowerPC architecture and Linux on PS3. Besides this, the Form history shows three e-mail addresses: rcmartel@uno.edu, rmartell@cs.uno.edu and fluidity@mail.com. Beside the Firefox history on the `nssal-linux-side-fs.dd` image, none of these e-mail addresses were found on the images. The browser was used on December 4th 2008, February 27th 2009 and March 10th 2009. The `jhuisi-linux-side-fs.dd` image contains very few Firefox traces of January 22nd 2009, it shows Adobe Flash was downloaded and CNN was visited.

[23]FoxAnalysis is available from: `http://forensic-software.co.uk/foxanalysis.aspx`
[24]MozillaCacheView is available from: `http://www.nirsoft.net/utils/mozilla_cache_viewer.html`

When the private data in Firefox is cleared, the SQLite files are not removed but the records in the database are overwritten with zeros [4]. Therefore it is not possible to recover complete Firefox SQLite history files from the unallocated space. However, it is possible to recover history from the unallocated space as for each browsing session a temporary `places.sqlite-journal` file is created which is removed when the browser is closed. This journal files contains visited URL's, website titles and bookmarks. However, the problem is that these files are hard to recover as they do not have a header or footer. By using the file system's journal it was possible to recover a `places.sqlite-journal` file on the `nssal-linux-side-fs.dd`. No `places.sqlite-journal` files could be recovered on the `jhuisi-linux-side-fs.dd` image. Furthermore, by comparing several `places.sqlite-journal` files we noticed that most of the stored URL's in the file are preceded by `0x01` and can be recovered by searching for ' `http://`'. Searching for this pattern will trigger some false positives, for example in the `/usr/bin/whois`. However, hits can be easily distinguished from relevant hits. URL's are grouped together, see Figure 12 for an example of some hits in WinHex in the `jhuisi-linux-side-fs.dd`. Furthermore, as `places.sqlite-journal` stores URL's of bookmarks it could be more effective to search for an unique bookmark that has been recovered with for example with FoxAnalysis. Unfortunately, no interesting URL's were extracted from the recovered `places.sqlite-journal` files.



Offset	Rel. ofs.	Search hits	Name	Ext.	Path
5125550336		01687474703A2F2F	Free space		Free space
5125550419		01687474703A2F2F	Free space		Free space
5125550454		01687474703A2F2F	Free space		Free space
5125550539		01687474703A2F2F	Free space		Free space

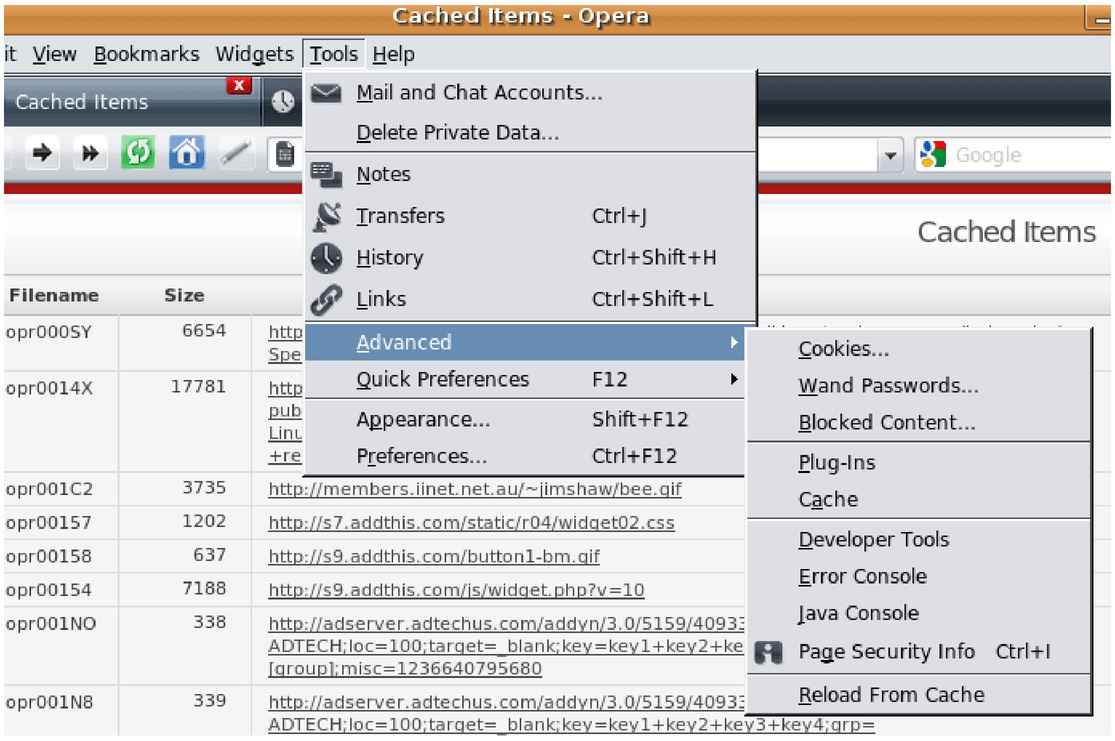
```
Offset
05196304384  22034,22082%26Values%3d1588%26Redirect%3d9..3.http://w
05196304442  nn.com/2..9.http://www.debian.org/...3.http://www.fsf.
05196304500  .!.G.http://www.ubuntu.com/support...C.http://www.ubun
05196304558  nux.org/.-._.http://www.ubuntulinux.org/wiki/FrontPage
05196304616  ..https://en-us.add-ons.mozilla.com/en-US/firefox/book
05196304674  s/.3.k.https://launchpad.net/distros/ubuntu/+addticket
05196304732  ▮a.place:folder=BOOKMARKS_MENU&folder=UNFILED_BOOKMARK
05196304790  lder=TOOLBAR&queryType=1&sort=12&excludeItemIfParentHa
```

Figure 12: Example of `places.sqlite-journal` entries

3.10.2 Opera

Opera makes use of the directory `/home/<user>/.opera/` to store its settings, cache and history. Opera itself is a good tool for examining its own traces. In order to examine the Opera history on the `nssal-linux-side-fs.dd` image, the `.opera` directory was copied from the image to an Ubuntu machine. Under the tools button in Opera it is possible to retrieve a wide variety of information: mail and chat accounts, notes, transfers, history, links, passwords, cookies, cache. Figure 13 illustrates the tools menu and the cached items viewer tool. The Opera history shows activity on 4th of December 2008, 21st of January, 26th of February, 5th, 6th and 9th of March 2009. All Opera history on the `nssal-linux-side-fs.dd` image is related to memory and kernel development.

Searching for a hexadecimal pattern '0x0a2d310a', which is found in the Opera history file `global.dat` to separate history entries, does not reveal any deleted browser history.



Filename	Size	
opr000SY	6654	http Spe
opr0014X	17781	http pub Linu +re
opr001C2	3735	http://members.iinet.net.au/~jimshaw/bee.gif
opr00157	1202	http://s7.addthis.com/static/r04/widget02.css
opr00158	637	http://s9.addthis.com/button1-bm.gif
opr00154	7188	http://s9.addthis.com/js/widget.php?v=10
opr001NO	338	http://adserver.adtechus.com/addyn/3.0/5159/40933 ADTECH;loc=100;target=_blank;key=key1+key2+ke [group];misc=1236640795680
opr001N8	339	http://adserver.adtechus.com/addyn/3.0/5159/40933 ADTECH;loc=100;target=_blank;key=key1+key2+key3+key4;grp=

Figure 13: Examaning the browser history with Opera.

3.11 Putting the gathered information together

The results of the investigation using the described methodology are presented in a time line added to this document as Appendix A. This graphical overview shows communications between the various systems. Each system has its own time line represented by a black vertical line. Events and communications between systems are both placed in the system time lines. The colors of the items, group the various items together, the meaning of the colors are indicated by the legend. The change, modification and access time of the drug recipes, the illicit images and the backdoor software are included. The conclusions of this investigation are made using the overview.

4 Conclusion

4.1 Relevant user activity

The relevant user activity is shown in appendix A containing a detailed time line of the user activity on both Linux images. This timeline shows data was exchanged between the two systems using the encrypted protocol SSH and time stamps of important events and relevant files. The bash history of the 'nssal' account on the suspect's Playstation was recovered. Besides this, terminal commands and output was recovered from physical memory of the suspects Playstation. The recovered bash history, the browser history and the log files show that the suspect was actively working/testing kernel/memory settings. Traces on the `nssal-thumb-fs.dd` appear to be shredded. Traces of backdoor software have been recovered from the Playstation of the suspect and log files show the system was compromised by using this software. The source code of the backdoor software and the compiled program was found on the JHUISI Playstation and network captures show that the backdoor was used on the JHUISI Playstation to gain access to the suspect's Playstation. Mardi Gras pictures and drug recipes were found on both the `nssal-linux-side-fs.dd` and `jhuisi-linux-side-fs.dd` image. The drug recipes and Mardi Gras pictures are equal on both images. On the `jhuisi-linux-side-fs.dd` image the directory with drug recipes also contains a sub directory 'customers' with a file that holds names and addresses. The network captures indicate that the suspect and the user of the JHUISI Playstation have met in the virtual Playstation Home environment.

4.2 Drug recipes

The drugs recipes on the `nssal-linux-side-fs.dd` image have an accessed and a modify time stamp between 11:49:40 and 11:50:05 on 2009-03-11, this is presumably the time when the files were created. Log files on the `jhuisi-linux-side-fs.dd` image shows that a SFTP session (session 3208) was created for the user 'goatboy' on 11:49:25 initiated from the `nssal-linux-side-fs.dd` and was closed on 11:50:09. The drug recipes were presumably copied from the JHUISI Playstation to the suspect's Playstation during this SFTP session, see Appendix A.

4.3 Mardi Gras pictures

The Mardi Gras pictures on the `jhuisi-linux-side-fs.dd` image were created between 11:52:00 and 11:52:04 on 2009-03-11. Log files on the `nssal-linux-side-fs.dd` image show that a SSH session (session 3266) was created for the user 'jhuisi' on 11:52:00 from the JHUISI Playstation and was closed on 11:52:06. Given the brief connection time of six seconds and the creation time of the Mardi Gras pictures during the connection, the images were presumably copied with SCP from the suspect's Playstation, see Appendix A.

The log files on the `jhuisi-linux-side-fs.dd` image show that the user 'goatboy' was logged in from the suspect's Playstation on the JHUISI Playstation through SSH (session 5237) between 11:45:22 and 12:14:16 on 2009-03-11. During this time frame the presumable transfer of the Mardi Gras pictures and drug recipes took place between the suspect's Playstation and the JHUISI Playstation, see Appendix A. During an SSH session it is possible to connect back to your own machine. Given the fact that it is possible to connect back to your own computer while one is remotely logged-in with SSH, both the suspect and the person on the JHUISI Playstation could have been responsible for transfer of the Mardi Gras pictures as these events took place while the user 'goatboy' was logged in from the suspect's Playstation. However, no traces have been found that the suspect connected back from the JHUISI Playstation to his/her own Playstation. This kind of activity leaves traces in the bash history of the 'goatboy' account on the JHUISI Playstation. No traces or indications have been found that the bash history of the 'goatboy' account was altered. Furthermore, the SSH activity leaves traces in the physical memory of the suspect's Playstation. Besides this, no `known_hosts` file is stored for the 'goatboy' account and no deletion traces of this `known_hosts` file were found which indicates that no SSH sessions have been initiated from the 'goatboy' account. Therefore, it is plausible that SSH connection with user 'goatboy' from the suspect's Playstation to the JHUISI Playstation was not used to connect back to his/her own Playstation, and that the suspect has therefore not initiated the transfer of the Mardi Gras pictures. However, no traces have been found on the `jhuisi-linux-side-fs.dd` image showing that a user from the JHUISI Playstation was responsible for the transfer of the Mardi Gras Pictures.

4.4 Backdoor

The recovered backdoor software on the `nssal-linux-side-fs.dd` image shows that the software appeared on the suspect's machine on 11:56:11 2009-03-11. Log entries on `nssal-linux-side-fs.dd` image show that an SSH session (session 3326) was created for the user 'jhuisi' on 11:56:09 initiated from the JHUISI Playstation and was closed on 11:56:11. Given the brief connection time of two seconds, the small size of the backdoor software and the creation time of the recovered backdoor and the source/compiled backdoor software on the JHUISI Playstation, the backdoor software was presumably transferred with SCP from the JHUISI Playstation. Log entries on `nssal-linux-side-fs.dd` shows that user 'jhuisi' was logged in with SSH (session 3208) from the JHUISI Playstation on 11:49:45 until 12:09:47 and presumably executed the backdoor without administrator privileges on 12:01:35. Log entries show that after the attempt of the user 'jhuisi' to execute the backdoor with administrator privileges, an SSH session for the user 'nssal' is created (session 3369) on 12:02:30 initiated from the JHUISI playstation and closed on 12:11:03. During this SSH session the user 'nssal' successfully executed the backdoor with administrator privileges, see Appendix A. It is possible that the suspect staged these events as he was logged in with user 'goatboy' on the JHUISI Playstation during the activity. However, as previously described, no traces have been found that this SSH connection was used to connect back to the suspect's Playstation. Besides this, the password of the user 'nssal' is 'nssal', therefore the user on the JHUISI account could have easily guessed the password, remotely logged in and executed the backdoor with administrator privileges. Furthermore, all backdoor events took place after the possible transfer of the drug recipes and the Mardi Gras pictures and are therefore likely not relevant for those activities.

4.5 Relation between suspect and JHUISI Playstation

It is not clear what the relationship between the suspect and the JHUISI Playstation is. Perhaps the 'goatboy' account on the JHUISI Playstation is used to exchange drug recipes. A customer directory with names and addresses was present and as described a SFTP session was presumably used to transfer the recipes from the JHUISI Playstation to the suspect's Playstation. It is not clear why the account 'jhuisi' was created on the suspect's Playstation, why a SSH session was created with user 'jhuisi' on 11:49:45 (session 3208) from the JHUISI Playstation to the Playstation of the suspect and if this session was authorized by the suspect. Log files on the `jhuisi-linux-side-fs.dd` show more short SSH activity that was initiated from the IP address '128.220.251.228' and '70.22.16.52' of which the purpose is unclear, see Appendix A.

5 Remarks

Throughout this report time is an essential part of the forensic investigation. In every forensic investigation time plays an important role. Therefore it would have been helpful if the investigators were more time accurate and for example registered how many seconds the clocks on the Playstations were out of sync with a trusted time source and at what time the file system images, the thumb drive image and memory dump were created.

Due to the limited amount of memory of a Playstation (256 MB) the installed Linux OS makes extensive use of the swap space. To aid the investigation it is recommended to include the swap space when creating a forensic duplicate: use `/dev/ps3da` instead of `/dev/ps3da1`. `/dev/ps3da` includes all partitions whereas `/dev/ps3da1` is only a single partition.

The administrator at the John Hopkins University was not able to create a memory dump. By providing a memory dump tool to the administrator and/or instructing him/her, valuable information could have been acquired.

Investigators should try to acquire as much information as possible, also non-technical information. Subsequently, all of this information should be provided to the digital forensic investigators as it can be very useful. For example, perhaps the administrator at the John Hopkins University was able to provide information such as who had physical access to the Playstation.

6 Further research

As described in section 3.3.2 determining mount options used to mount a file system can aid digital forensic research. Further investigation and the acquisition of more data is needed to be able to get founded and final conclusions. Especially the difference between a "cleanly" installed reference system and systems running for longer times of an OS version needs attention. Another method could be to find the part of the file system where access is frequent and check there for time stamp trends. The conclusions could result in a set of values typical for the different mount option and allow forensic tools to be able to determine where to watch out for when a ext3 file system or any other UNIX file system is examined.

7 Acknowledgments

We would like to thank the employees of the Open Systems group in the department of Digital Technology and Biometrics of the Netherlands Forensic Institute for their support. A special thanks to Ruud van Baar and Feico Dillema for their constructive comments and suggestions.

8 Bibliography

References

[1] Brian Carrier. *File System Forensic Analysis.* Addison-Wesley Professional, 2005.

[2] Brian Carrier. Why recovering a deleted ext3 file is difficult..., 2005. `http://linux.sys-con.com/node/117909?page=0,1`.

[3] Alex J. Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.

[4] M.T. Pereira. Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records. *Digital Investigation*, 5(3-4):93–103, 2009.

[5] Playstation.com. Playstation network - home, July 2009. `http://www.us.playstation.com/PSN/Home#fbid:JEJaSHcxgXT`.

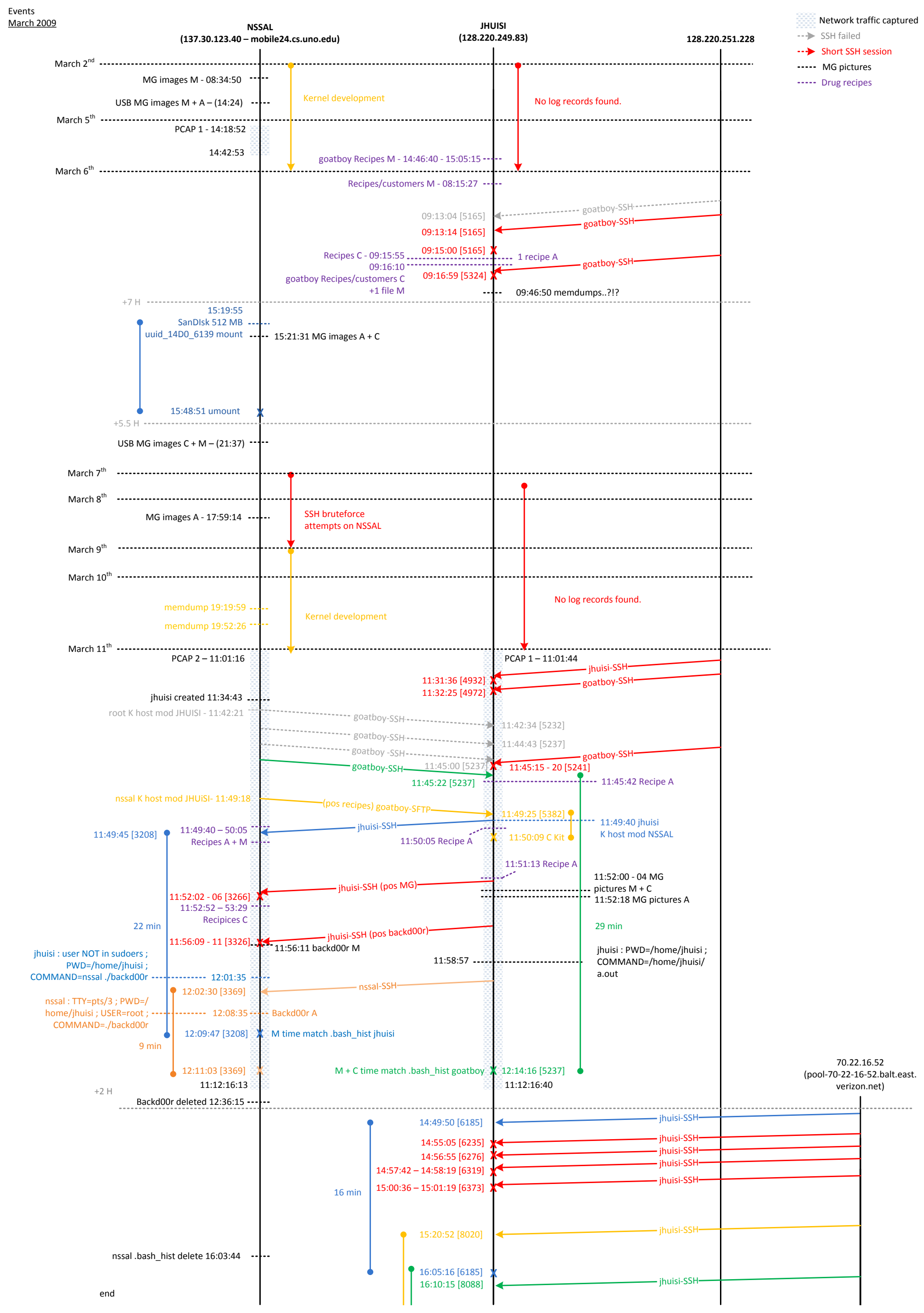
[6] Wikipedia.org. Secure copy, July 2009. `http://en.wikipedia.org/wiki/Secure_copy`.

[7] Wikipedia.org. Ssh file transfer protocol, July 2009. `http://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol`.

[8] Carlo Wood. Howto recover deleted files on an ext3 file system, 2008. `http://www.xs4all.nl/~carlo17/howto/undelete_ext3.html`.

9 Appendix A - Overview

Click on the image to magnify.



Events
March 2009
NSSAL
(137.30.123.40 – mobile24.cs.uno.edu)
JHUISI
(128.220.249.83)
128.220.251.228
Network traffic captured
SSH failed
Short SSH session
MG pictures
Drug recipes
March 2nd
MG images M - 08:34:50
USB MG images M + A – (14:24)
Kernel development
No log records found.
March 5th
PCAP 1 - 14:18:52
14:42:53
goatboy Recipes M - 14:46:40 - 15:05:15
March 6th
Recipes/customers M - 08:15:27
goatboy-SSH
09:13:04 [5165]
goatboy-SSH
09:13:14 [5165]
09:15:00 [5165]
Recipes C - 09:15:55
1 recipe A
09:16:10
goatboy-SSH
goatboy Recipes/customers C
09:16:59 [5324]
+1 file M
09:46:50 memdumps..?!?
+7 H
15:19:55
SanDIsk 512 MB
uuid_14D0_6139 mount
15:21:31 MG images A + C
15:48:51 umount
+5.5 H
USB MG images C + M – (21:37)
March 7th
March 8th
MG images A - 17:59:14
SSH bruteforce
attempts on NSSAL
March 9th
March 10th
No log records found.
memdump 19:19:59
Kernel development
memdump 19:52:26
March 11th
PCAP 2 – 11:01:16
PCAP 1 – 11:01:44
jhuisi-SSH
11:31:36 [4932]
goatboy-SSH
11:32:25 [4972]
jhuisi created 11:34:43
root K host mod JHUISI - 11:42:21
goatboy-SSH
11:42:34 [5232]
goatboy-SSH
11:44:43 [5237]
goatboy -SSH
goatboy-SSH
goatboy-SSH
11:45:00 [5237]
11:45:15 - 20 [5241]
11:45:22 [5237]
11:45:42 Recipe A
nssal K host mod JHUISI- 11:49:18
(pos recipes) goatboy-SFTP
11:49:25 [5382]
11:49:40 jhuisi
K host mod NSSAL
11:49:45 [3208]
11:49:40 – 50:05
jhuisi-SSH
Recipes A + M
11:50:05 Recipe A
11:50:09 C Kit
11:51:13 Recipe A
11:52:00 - 04 MG
pictures M + C
jhuisi-SSH (pos MG)
11:52:02 - 06 [3266]
11:52:18 MG pictures A
11:52:52 – 53:29
Recipices C
22 min
29 min
jhuisi-SSH (pos backd00r)
11:56:09 - 11 [3326]
11:56:11 backd00r M
jhuisi : user NOT in sudoers ;
PWD=/home/jhuisi ;
COMMAND=nssal ./backd00r
jhuisi : PWD=/home/jhuisi ;
COMMAND=/home/jhuisi/
a.out
11:58:57
12:01:35
12:02:30 [3369]
nssal-SSH
nssal : TTY=pts/3 ; PWD=/
home/jhuisi ; USER=root ;
COMMAND=./backd00r
12:08:35
Backd00r A
12:09:47 [3208]
M time match .bash_hist jhuisi
9 min
12:11:03 [3369]
M + C time match .bash_hist goatboy
12:14:16 [5237]
11:12:16:13
11:12:16:40
70.22.16.52
(pool-70-22-16-52.balt.east.
verizon.net)
+2 H
Backd00r deleted 12:36:15
14:49:50 [6185]
jhuisi-SSH
14:55:05 [6235]
jhuisi-SSH
14:56:55 [6276]
jhuisi-SSH
14:57:42 – 14:58:19 [6319]
jhuisi-SSH
15:00:36 – 15:01:19 [6373]
jhuisi-SSH
16 min
15:20:52 [8020]
jhuisi-SSH
nssal .bash_hist delete 16:03:44
16:05:16 [6185]
16:10:15 [8088]
jhuisi-SSH
end