VANJA KORAĆ
Mathematical Institute SASA
Belgrade, Serbia
vanja@turing.mi.sanu.ac.rs

004.451.9LINUX:902
COBISS.SR-ID 212300300

Original research article
Received: December 04th 2013
Accepted: February 15th 2014

# DIGITAL ARCHAEOLOGY OF VOLATILE DATA ON A LINUX PLATFORM

## ABSTRACT

*It is necessary to gather, analyse and store data on a "live" system in order to detect, in time, whether there is incident/illegal activity taking place. Since it is very important to collect volatile data, in this paper ways of collecting are described as well as tools which are related to them on a Linux platform. In the paper, the following volatile data are named which are gathered from compromised systems:*

*System time and date, the existing network connexions, open TCP and UDP ports, executive files which open TCP and UDP ports, processes and services started, opened files, internet routing and cache tables, read modules and the kernel, memory and memory process content and mounted system file.*

**Keywords:** digital archaeology, forensic analysis, data collecting, Linux system, "live" collecting, volatile data, compromised system.

Linux is a free UNIX orientated open source operating system, originally created by Linus Torvaldsa with the assistance of programmers from all over the world. There are different Linux versions which are called distributions, with the most common ones being Red Hat, Centos, Fedora, Debian, Suse, Ubuntu, Kubuntu and Slackware. Each of them possesses advantages and disadvantages. These distributions differ from each other also according to their types. There are server, desktop and live distributions. Server distributions are primarily orientated towards the business environment (although they can be configured for home usage). Desktop distributions are more appropriate for home usage; which needs a graphical environment and a great number of applications. Live distributions need bootable versions of operating systems which are read directly into RAM memory and run independently from the existing computer operating system. What forensic investigators need to know is the way in which Linux boots its operating system. The Linux boot sequence starts by reading the kernel. By default, the kernel image is usually situated in the boot directory. Further on, the link to the kernel image is situated in the boot directory and is referenced from the file of the Linux loader, LILO (/etc/lilo) or GRUB (/etc/grub.conf). The last step is initialisation. Files controlling initialisation are situated in the /etc/inittab file. The file responsible for starting the process is /sbin/init. After that, the run level is initialised and the start up scripts are controlled by the terminal process. When it comes to the Linux file system, it is necessary to highlight that Linux treats all the devices as files and stores them in folder/dev (Nelson, Phillips and Steuart 2010). For forensic inves-

tigators it is important to know that most of the Linux distributions possess organised files with similar directory structures:

| | |
|---|---|
| /bin | Common executive commands at system level |
| /boot | Files necessary for booting systems including kernel images together with links indicating them defined in LILO or GRUB |
| /usr | Local programs, libraries, games |
| /var | Logs and other variable files |
| /dev | Interface files which enable the kernel to communicate with hardware and the file system |
| /home | Directories of all of the users on the system with personal user and configuration files |
| /mnt | Mount points for external, remote and portable file systems |
| /etc | Configuration files and scripts for administration |
| /root | Directory of the root user |
| /sbin | Administrative executive commands which should be accessible only to root, i.e. administrator |
| /lib | Basic system libraries |
| /opt | Optional and other programs |

Table 1 Linux directory structure

Basic commands which can be of assistance to forensic investigators for gaining basic information about the investigated system are as follows:

#uname - a – shows the name of the computer and the Linux version
#ls – shows the file list
#ls -l – shows the file list with their permissions
#ls -ul file name – gives the time of access to the file
#cp – copies files
#mv – moves files
#chmod – permission changes of files
#ps – shows started processes
#netstat -s – shows information and protocols
#ifconfig – shows information about network devices on the system
#find – for searching for information on the system
#grep – for searching for files or searching with key words
#less – lists file content
#more - lists file content
#cat – also lists file content
#diff – compares two files
#df – shows mounted file systems

In order to use file systems on Linux, it needs to be mounted on a system. All of the file systems on partitions which are defined during installing Linux operating system will be automatically mounted during each system boot. Forensic investigators need to know that data can be written to the device even though the device itself is not mounted. Periphery devices for data storage are recognized as SCSI devices (Nelson, Phillips and Steuart 2010). If an IDE disc is used on the primary controller as the master, the system will name it “hda”, but if it is mounted as a slave, it will be named “hdb”. On the secondary controller, it will be “hdc” or “hdd”. In order to see the complete list of partitions available on a system, the following command is used:

#fdisk -l /dev/hda

Each partition has its ascribed Linux name. The “*” mark means that it is a bootable partition. Exit of the fdisc command also includes information about the initial and last cylinder of each partition, as well as the number of blocks they contain, the ID of the partition and the type of file system.

During an investigation, forensic investigators need to ascertain the certain precautionary measures, such as:

- avoiding program initiation on a compromised computer system
- not initialising programs which can change metadata about files and directories
- document all of the activities undertaken and results gained during the investigation
- calculating hash data values in order to secure data integrity

In order to preserve data in a "live" system, it is often necessary to establish whether there is an incidental/illegal action. Malicious programs can, for example, endanger the security of a system itself as well as the security of any system to which it is connected. Due to the importance of gathering volatile data, further on in this paper the methods and tools will be described which deal with this process on a Linux platform. Just like forensic investigations of a Windows system, for a forensic investigator it is necessary to possess his/her own "set" of tools for the collection of volatile data from a compromised system. The reason is that certain commands on a compromised system can also be endangered and, therefore, they cannot be treated as reliable and, as a result, interaction with the system being investigated is reduced to minimum. By using personal tools, it is possible to discover valuable data which are hidden by certain malicious programs (rootkit for example). Certainly, in some cases when one is dealing with a rootkit which is read as a loadable kernel module (LKM), these tools will not give the expected results and it will be necessary to perform a forensic investigation of the memory and the file system. This chapter offers a general methodology for preserving volatile data on a Linux platform in a correct forensic way, by giving certain practical examples and pointing out the advantages and disadvantages of collected data, as well as its influence on system security.

Linux possesses a suitable tool which can record started commands and their exits, which makes it easy to document what was performed on a live system. The tool is called "Script" and it caches data in the memory and records all the information when it is interrupted in file typescript. If it is required to record after each command, then the script tool is used with an "-f" switch.

#script or #script -f

Before starting any of the commands on an investigated Linux system from a reliable command shell[1] , the script tool is started (previously compiled in a reliable environment). On a forensic system (which can be either a Linux or Windows working station) netcat actually cryptcat command[2] can be started (a tool which can collect data from a network and recommended by many experts) in the following way:

#netcat -v -l -p 9898 > exit_commands_from_compromised_computer.txt

Data sent over the 9898 port onto a forensic station will be saved in the file exit_commands_from_compromised_computer.txt. On an investigated computer (Linux), commands will be started with which data of importance for forensic investigation will be collected and their exits will be sent to a forensic working station over the 9898 port:

#/mnt/cdrom/command | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

With the CTRL-C command, the netcat session is interrupted.

What needs to be done next is to generate the hash value over the obtained data (MD5, SHA-1 or SHA-256).

Further on in the paper, it will be shown how volatile data are collected which are of extreme importance for a forensic investigation. Volatile data includes:

a. System time and date
b. Existing network connections
c. Opened TCP and UDP ports
d. Executive files that open TCP and UDP ports
e. Initiated processes
f. Opened files
g. Internet routing table
h. loadable kernel module LKM
i. mounted file systems

1 Shell represents an interface between the core of an operating system and the user. It possesses a great number of functions, among which the interpreting command line, starting programs, in- and output direction, pipe and shell programming stand out. The most famous shell command interpreters are Bourne shell (sh), C shell (csh), Bourne-again shell (bash), Korn shell (ksh)

2 If it is desired to send data from the investigated computer to a forensic working station, Netcat can be used with an encrypt called cryptcat. Cryptcat uses an improved version of the Twofish blockcoded encrypt with a symetric key. More about this king of encrypt can be found on the Schneier website: http://www.schneier.com/twofish.html

**Important volatile data on Linux - System time and date**

The system time and date can be obtained by using a command which also marks the beginning of forensic time collecting:

#date

an exit from this command is: Sun Feb 3 13:54:31 CET 2013

By using the forensic netcat command, it is done as follows:

#netcat -v -l -p 9898 > date_of_compromised_computer

#/mnt/cdrom/date -u | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum date_of_compromised_computer > date_of_compromised_computer.md5

After finishing the forensic collecting, it is recommended also to mark the time of finishing data collecting from the compromised computer.

#netcat -v -l -p 9898 > time_end_of_compromised_computer

#/mnt/cdrom/date -u | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum time_end_of_compromised_computer > time_end_of_compromised_computer.md5

**Important volatile data on Linux – Existing network connections**

These data are important activity indicators on an investigated system. According to these obtained data, forensic investigators can find out whether a malicious user is still connected to the system used by the port. It is also possible to find out the initial point of interruption, the enabled vulnerable services on the system which could have been compromised and intruded into the system. A command which can list the existing network connections is the netstat command shown on figure 1.

```
root@ubunt1104srvx86:~# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:995             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
```

Fig. 1 Image of the exiting command netstat -an

#netstat -an

**Important volatile data on Linux - Opened TCP and UDP ports**

The forensic investigation of the opened ports (TCP and UDP[3] ) on a system is focused on detecting vulnerable ports or backdoors established on a system which make incident/illegal actions possible. In figure 2 a command is shown which depicts not only the ports of IP addresses but also the names of processes and their Ids which are responsible for the opening of a certain port (Prosise and Mandia 2001):

#netstat -plant

3 It can be expected that on a system, next to the TCP and UDP ports there can also be a RAW port. It is necessary to know that it is related to the Linux kernel

```
root@ubunt1104srvx86:~# netstat -plant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN
619/dovecot
tcp        0      0 0.0.0.0:995             0.0.0.0:*               LISTEN
619/dovecot
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
733/mysqld
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN
619/dovecot
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN
619/dovecot
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
1185/apache2
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
941/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
892/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
743/cupsd
```

Fig. 2 Image of the exiting netstat –plant command

By using netcat, forensic investigators can collect this information on a "live" system in the following manner [Kruse and Heiser 2010]:

#netcat -v -l -p 9898 > open_port_TCP_UDP_of_compromised_computer

#/mnt/cdrom/netstat -plant | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum open_port_TCP_UDP_of_compromised_computer > open_port_TCP_UDP_of_compromised_computer.sh256

The best form of protection from opened ports (since they bear potential risks of intrusion into the system) is to open only the ports which are necessary for correct system function. It is also recommended to exclude services that work on unnecessary ports in order to increase the security of the system.

### Important volatile data on Linux – Executive files that open TCP and UDP ports

It is a Linux tool which can link the opened port with the started process lsof[4] (List Open Files), giving the list of active processes. One of the features of this tool is that it not only shows the process which opens a certain port, but it also shows the files which initiate certain processes. In cases when a malicious user compromised the system and transferred certain malicious files, he/she will try to hide these files from the system by marking them as hidden. Also, the malicious user will try to create a process which, after opening the file, disconnects the link with the file (unlinks), while the process continues to perform malicious activities.

Programs of the "ls" type will not display this information about the file and the process, since they are hidden from the administrator. It is therefore very important that the forensic investigator or administrator knows the limits of the programs they are using. Lsof is a program that offers detailed information about files, including also files with disconnected bonds. What needs to be highlighted is that knowing the available tools and choosing the right one is of utmost importance for forensic investigation as well as for establishing a safe system.

It can be used as:

#lsof -n , giving a detailed image of the files, processes and ports on the system, but the investigation can be reduced to the processes that are related to TCP and UDP internet sockets shown on figure 3 with the command "#lsof -i".

4 Available at Purdue University: http://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/lsof/

By using this command in a forensically correct way, all information about all of the processes on the system, opened ports and files can be collected. It is as follows:

#netcat -v -l -p 9898 > lsof_of_compromised_computer

#/mnt/cdrom/**lsof -n -P -l** | /mnt/cdrom/netcat ip_addressa_of_forensic_working_station 9898

#sh256sum lsof_of_compromised_computer > lsof_of_compromised_computer.sh256

INETD is a process that steers standard Internet services on a system. It starts when a system is booted and it uses a configuration file in which it is defined which service will make it run. The main configuration file inetd uses /etc/inetd.conf (place and name depend on the Linux distribution type). For a secure system it is important to understand the way inetd works and which information in the configuration file it contains (Cole 2002).

The second way is to start the "top" command:

```
root@ubunt1104srvx86:~# lsof -i
COMMAND    PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae  385    avahi   13u  IPv4   6655      0t0  UDP *:mdns
avahi-dae  385    avahi   14u  IPv6   6656      0t0  UDP *:mdns
avahi-dae  385    avahi   15u  IPv4   6657      0t0  UDP *:44263
avahi-dae  385    avahi   16u  IPv6   6658      0t0  UDP *:38650
smbd       476     root   24u  IPv6   7615      0t0  TCP *:microsoft-ds (LISTEN)
smbd       476     root   25u  IPv6   7617      0t0  TCP *:netbios-ssn (LISTEN)
dovecot    592     root    6u  IPv4   7673      0t0  TCP *:imap2 (LISTEN)
dovecot    592     root    7u  IPv4   7674      0t0  TCP *:imaps (LISTEN)
dovecot    592     root    8u  IPv4   7675      0t0  TCP *:pop3 (LISTEN)
dovecot    592     root    9u  IPv4   7676      0t0  TCP *:pop3s (LISTEN)
mysqld     703    mysql   10u  IPv4   7979      0t0  TCP localhost:mysql (LISTEN)
cupsd      741     root    5u  IPv6   7943      0t0  TCP ip6-localhost:ipp (LISTEN)
cupsd      741     root    6u  IPv4   7944      0t0  TCP localhost:ipp (LISTEN)
```

Fig. 3 Image of the exiting command lsof -i

It should be highlighted that data obtained with netstat and lsof commands should be compared to each other, because only in their differences can a forensic investigator find hidden processes from the kernel with a LKM. This paper will not go into details about LKM, since it forms a separate field of investigation and can be discussed in other papers. Spotting suspicious ports by using a digital forensic investigator can offer additional data about incident/illegal activity, while by using the administrator it is possible to stop further damage to the system.

**Important volatile data on Linux - Initiated processes and services**

In order to keep the system safe, one needs to know which processes and services are initiated on it. In Linux, initiated processes can be detected in several ways. One of them is the usage of the "Ps" command (Prosise and Mandia 2003):

#ps -auxwww, lists all the processes on the system and the users who started them.

#top

The top program will generate the whole screen with a list of existing processes which are permanently updated according to the degree of usage of the CPU. On top of the list there are data about time during which the operating system has been running, the number of processes run on the system and statistics about the available memory and swap space. The image can be formed in different ways with the "SHIFT+o" buttons, so that processes can be sorted according to their CPU (SHIFT+p) space, memory (SHIFT+m) and swap and the id of the process etc. In order to mark the activated process more easily, the "z" button is used, which colours all the processes red, while the most intense process, according to given parameters, is marked white (Figure 4). It is also possible to show started processes by a specific user:

#top -u of suspicious_user

What is of importance to a forensic investigator is the image of the absolute paths of running processes, which is performed with the "c" button after running the top command. There was an actual case of this. The http server of an organisation suddenly stopped working. After analysing data on the server it was realised that there were no attacks, but the Zope server was updated with bugged files and it generated a great amount of log files which occupied 100% of the disc space. After analysing the top command, a process was enabled which generated malfunctions and also the file which wrote these malfunctions. After stopping this process, deleting the overloaded log file and correcting the bugged files, the Zope http server was successfully restarted and the problem was solved.

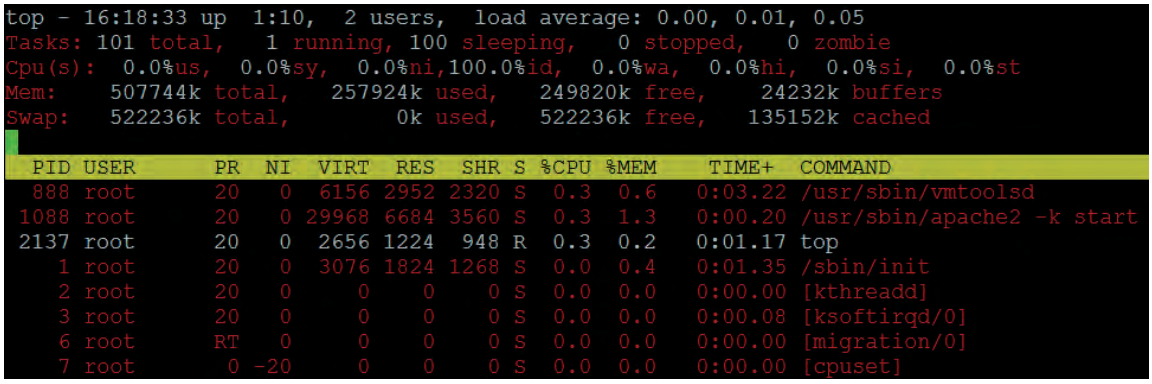


Fig. 4 Image of exiting top command

Services run on a system can also be identified with the command:

#service --status-all

#ps -A

**Important volatile data on Linux – Opened files**

By using the lsof tool, a list of opened files on a system can be obtained. These can represent valuable data for a forensic investigation. For example, hidden files can be revealed, such as malicious tools which can be password crackers, exploits and other malicious programs which use resources of the server itself to act against either users of the system or against other systems for the distribution or hosting of forbidden content, the distribution of content protected by copyright and many other kinds of highly technological crimes.

Commands which can be of use to a forensic investigator are as follows:

#lsof -u root shows all of the opened processes and files opened by the root user

#lsof -p 3333 shows all of the opened files by the process with ID 3333

#lsof /var/log/auth.log shows processes which are opened with a certain file as shown in Fig. 5.

#lsof /home can figure as very useful information if a system shows devices or resources as busy, since it can show which processes are responsible for mounting the spot /home on a system.

**Important volatile data on Linux - Internet routing table and cache tables**

In order to enable an administrator or a forensic investigator to realise whether the routing table was changed, a netstat command (or route command) is used, due to the same reasons as described in the chapter describing volatile data on Windows. It is shown in Fig. 6:

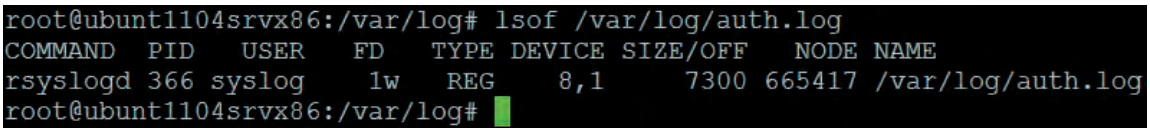


Fig. 5 Image of an exit of the lsof command over the file /var/log/auth.log

#route
or
#netstat -nr

Address Resolution Protocol, or ARP, is a TCP/IP protocol used to turn an IP address into a physical address, actually a MAC address. For a digital investigator it is of importance to examine the ARP cache of the examined computer, since it is possible to identify other systems which are currently making or recently made a connection to an examined computer. According to that, information gathered via the ARP cache is used to discover additional computers on the network which could be compromised as a result of incident/illegal activities. From the security point of view, examining the ARP cache is used to identify suspicious computer systems on the network which can be used for initiating internet attacks on the network. In order to display the content of an ARP cache on a system, the "arp" command is used (Fig. 7):

#/mnt/cdrom/arp

This command will display all IPs which are connected or were connected to the examined computer.

By using netcat, forensic investigators can collect these data on a live system as follows:

#netcat -v -l -p 9898 > ARPcache_of_compromised_computer
#/mnt/cdrom/arp | /mnt/cdrom/netcat ip_address_of_forensic_work_station 9898

#sh256sumARPcache_of_compromised_computer>ARPcache_of_compromised_computer.sh256

Currently active routes on a system are kept at the so-called routing cache, i.e. the kernel route cache table. It should be mentioned that the route cache table is different from a route table. The kernel route cache table shows currently active (connected) routes on a system. The route table is used for making decisions about routing, while the kernel route cache table displays routes which are connected. Exactly this can be of great importance to a forensic investigator, since collecting data for a kernel route cache table can help discover computers on the network which could be compromised, but also computers from which an incident/illegal activity was performed. From a security point of view, collecting information from the route cache table can be used for identifying suspicious computer systems on the network which can be used to initiate attacks within the network. The kernel route cache table shows sources and destinations, the gateway and the interface which is used to make the connection.

```
root@ubunt1104srvx86:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
```

Fig. 6 Image of the exit command netstat -rn

On Linux systems, examination of the kernel route cache can be achieved with the "route" command (Fig. 8):

#route -Cn

By using netcat, a forensic investigator can collect information on a live system as follows:

#netcat -v -l -p 9898 > routing_cache_table_of_compromised_computer

```
[root@turing ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
147.91.96.190            ether   a0:f3:c1:a2:0c:05   C                     eth0
147.91.96.208            ether   a0:f3:c1:a2:41:b3   C                     eth0
147.91.96.148            ether   00:19:99:d3:90:33   C                     eth0
147.91.96.149            ether   00:19:99:e4:cf:d6   C                     eth0
147.91.96.146            ether   50:26:90:a1:46:f6   C                     eth0
```

Fig. 7 Display of the ARP cache with an arp command

```
root@emis:~# route -Cn
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref    Use Iface
66.249.76.226   147.91.102.6    147.91.102.6    l     0      0       49 lo
147.91.102.6    66.249.76.162   147.91.102.1          0      0        1 eth0
147.91.102.6    66.249.76.226   147.91.102.1          0      0        7 eth0
147.91.102.6    66.249.76.14    147.91.102.1          0      0       14 eth0
147.91.102.6    147.91.96.2     147.91.102.1          0      2        0 eth0
147.91.102.6    147.91.96.2     147.91.102.1          0      0        4 eth0
192.168.2.4     192.168.2.255   192.168.2.255   ibl   0      0       11 lo
```

Fig. 8 Image of the content of a kernel route cache table

```
root@ubunt1104srvx86:~# lsmod
Module                  Size  Used by
vesafb                 13449  1
snd_ens1371            24722  0
gameport               15027  1 snd_ens1371
snd_rawmidi            25269  1 snd_ens1371
snd_seq_device         14110  1 snd_rawmidi
snd_ac97_codec        105614  1 snd_ens1371
ac97_bus               12642  1 snd_ac97_codec
snd_pcm                80244  2 snd_ens1371,snd_ac97_codec
ppdev                  12849  0
vmw_balloon            12729  0
psmouse                59039  0
snd_timer              28659  1 snd_pcm
```

Fig. 9 Image of modules currently read into the kernel with the lsmod command

#/mnt/cdrom/route -Cn | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sumrouting_cache_table_of_compromised_computer>routing_cache_table_of_compromised_computer.sh256

**Important volatile data on Linux - Modules read into LKM**

If a forensic investigator has reasonable suspicions that the existing kernel of an investigated computer is compromised with a certain rootkit i.e. Trojan, it would be necessary to list modules read into the kernel. This can be done in the following ways:

#netcat -v -l -p 9898 > modules_of_compromised_computer

#/mnt/cdrom/cat /proc/modules | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum modules_of_compromised_computer > modules_of_compromised_computer.sh256

or with an approved command lsmod:

#netcat -v -l -p 9898 > modules_of_compromised_computer

#/mnt/cdrom/lsmod | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum modules_of_compromised_computer > modules_of_compromised_computer.sh256

Exit of the lsmod command is shown in Fig. 9:

What a forensic investigator has to bear in mind is that there are practical techniques which allow malicious modules to be read into the kernel and hide it afterwards. One such rootkit is called Knark[5] (more information about Knark can be found on the SANS[6] website). After hiding the read module, there is no possibility to discover it with a “live” forensic investigation.

5 Available at http://www.sans.org/security-resources/idfaq/knark.php

6 http://www.sans.org/

**Important volatile data on Linux - Dump memory and memory processes**

In order to analyse the memory of a compromised computer, a forensic investigator needs to capture it physically. One should bear in mind that when a memory dump is performed, the current state of the memory is disturbed by initiating programs and reading data. An additional problem exists with the writing of a file with the captured state of the physical memory. It means that any exiting file will be cached in memory, possibly replacing very important information which might be valuable for further digital investigation. This is why using a forensic computer is the best way to save data with minimum influence on memory. Actually, when it comes to capturing memory, a forensic investigator faces a dilemma: the aim is to preserve as much very changeable data as possible, but by collecting it, additional proof can be destroyed. The decision n investigator faces requires him to ascertain whether the importance of the collected data is greater than the importance of data that will be lost. This decision relies greatly on the experience of the forensic investigator.

Since it is obligatory to document the entire procedure, it is recommended to collect basic data about the memory. The file containing this data is /proc/meminfo. This is done as follows:

#netcat -v -l -p 9898 > mem_info_of_compromised_computer

#/mnt/cdrom/cat < /proc/meminfo | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum mem_info_of_compromised_computer>mem_info_of_compromised_computer.sh256

The simplest way, although not universal, for capturing the complete physical memory on Linux systems is to initiate the approved static compiled dd command.[7]

#netcat -v -l -p 9898 > fiz_mem_of_compromised_computer

#/mnt/cdrom/dcfldd < /dev/mem | /mnt/cdrom/netcat ip_addess_of_forensic_working_station 9898

#sh256sum fiz_mem_of_compromised_computer > fiz_mem_of_compromised_computer.sh256

This process functions on Linux systems, but some Unix systems (for example FreeBSD, Solaris) treat physical memory in a different manner, which can result in incomplete content of physical memory (Farmer and Venema 2008). There is a "memdump" tool within The Coroner's Toolkit-a (TCT)[8] which successfully solves the mentioned problem using minimum memory with minimum influence on it. The correct way of applying this program is as follows:

#netcat -v -l -p 9898 > fiz_mem_of_compromised_computer

#/mnt/cdrom/memdump | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum fiz_mem_of_compromised_computer > fiz_mem_of_compromised_computer.sh256

The content of physical memory can also be accessed from file /proc/kcore. This file contains data from the physical memory which are in the ELF[9] core file format. There is an accepted opinion that it is highly recommended to collect the content of this file along with raw format data from the memory. The reason is that this format can be examined with the GNU debugger, the so-called GDB[10] with the help of the "System map" file and kernel image from the /boot directory. This process was described by Mariusz Burdach in one of his papers[11] (Burdach 2004).[12]

When it comes to memory, swap space in a system is also of forensic importance. Swap space represents space in which pages are temporarily stored in cases where the freeing up part of the RAM memory is needed[13] (or the system needs more memory than available in the existing RAM). The total

7 http://dcfldd.sourceforge.net/

8 The Coroner's Toolkit (TCT) represents a collection of forensic tools whose authors are Wietse Venema and Dan Farmer, available at http://www.porcupine.org/forensics/tct.html , 05.04.2013

9 ELF - Executable and Linking Format

10 https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/s2-proc-kcore.html

11 http://www.symantec.com/connect/articles/detecting-root-kits-and-kernel-level-compromises-linux

12 http://www.symantec.com/connect/articles/forensic-analysis-live-linux-system-pt-2

13 Kernel can move onto the swap space those parts of memory which are used less (inactive) and to ascribe the free memory to a current program i.e. process which needs memory

```
root@ubunt1104srvx86:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1             20125340   2225092  16877940  12% /
none                    247256       200    247056   1% /dev
none                    253872         0    253872   0% /dev/shm
none                    253872       444    253428   1% /var/run
none                    253872         0    253872   0% /var/lock
```

Fig. 10 Exit of the df command displaying mounted devices

of RAM memory and swap memory represents the total of virtual memory on a system. Swap space can exist in the form of a swap partition (recommended), a swap file or a combination of both a swap file and a partition (Volonino, Anzaldua and Godwin 2007). This space can contain important information for a forensic investigation (although the system rarely swaps). This space can simply be copied by using the "dd" or "cat" tools over the swap partition or file with the help of the "netcat" tool and later be searched with tools looking for certain strings (for example hexdump[14] ).

A memory dump of initiated processes can be done by using the tool pcat which is in the already mentioned The Coroner's Toolkit-a (TCT)[15] The forensically correct usage of this command is:

#netcat -v -l -p 9898 > pcat_of_compromised_computer

#/mnt/cdrom/pcat proc_id | /mnt/cdrom/netcat ip_address_of_forensic_working_station 9898

#sh256sum pcat_of_compromised_computer > pcat_of_compromised_computer.sh256

For the analysis of the memory itself, it is of great importance to know the way the system uses memory (caching files and virtual memory pages, which aim to improve computer performance). From all that was said above it is clear that important memory parts can be found and identified on a system, which can be of great use for digital investigation.

### Important volatile data on Linux – Mounted file systems

For a forensic investigation it is important to find out which file systems were mounted in the examined operating system. There are certain commands which enable this. The first one is the mount command:

#mount whose exit displays devices (the hard disc for example), mounting spot and the type of file system.

The second one is the df command:

#df whose exit displays mounted devices (Fig. 10), the mounting spot, the size and available capacities and how much is occupied.

The previous two commands cannot identify network file systems or the NFS. The command which is able to display NFS shared resource is "showmount":

#showmount -a localhost or showmount which displays exported systems

**#showmount -a localhost**
**All mount points on localhost:**
**192.168.1.104:/nfs/images**
**192.168.1.101:/nfs/films**

It shows a list of computer systems which are connected on a local system and their access points.

This command can enable forensic investigators to collect valuable data from an examined operating system, especially when it comes to the unauthorised distribution of copyrights or forbidden pornographic content.

### Final discussion

The essence of digital archaeology of volatile data is collecting data (potential proof) which will confirm or deny the existence of certain incident or illegal activity. In cases of incident or illegal activity, a decision needs to be made whether to turn off and remove the computer from the network and only then collect potential proof or if digital forensics will be performed on a live system. Such a decision does not only de-

14 http://linux.about.com/library/cmd/blcmdl1_hexdump.htm

15 http://www.porcupine.org/forensics/tct.html

pend on incident or illegal action or activity, but also on the types of systems themselves, for example, complex critical systems of banks or electronic management. Conditions in which there is a need for a "live" investigation appear more and more often. The "live" investigation of a computer system needs to be structured in such a way that it is focused and performed quickly and efficiently by experts. Since there is great importance attached to the collection of volatile data, the methods of collection and the tools used are described in the paper related to collecting on a Linux platform. The following volatile data were collected: system time and date, the existing network connections, open TCP and UDP ports, executive files which open TCP and UDP ports, processes and services started, opened files, internet routing and cache tables, read modules and the kernel, memory and memory process content and mounted system file. In accordance with the named goal, this paper is orientated to collecting data on a "live" Linux system from exactly determined spots on a system. Potential proof can be found on these spots which can, on one hand, indicate forensically relevant actions which influence system security and, on the other hand, confirm or deny the existence of certain incidents i.e. illegal activity.

## BIBLIOGRAPHY

**Cole, E. 2002**
*Hackers Bewar*, New Riders Publishing.

**Farmer, D. & Venema, W. 2008**
*Forensic discovery*, Pearson Education Inc, Crawfordsville.

**Burdach, M. 2004**
*Forensic Analysis of a Live Linux System, Pt. 2, Symantec*, April 2004 (http://www.symantec.com/connect/articles/forensic-analysis-live-linux-system-pt-2).

**Nelson, B., Phillips, A. & Steuart, C. 2010**
*Guide to Computer Forensics and Investigations – fourth edition*, Course Technology, Cengage Learning, Boston.

**Volonino, L., Anzaldua, R. & Godwin, J. 2007**
*Computer forensics principles and practices*, Pearson Education, Inc Upper Saddle River, New Jersey.

**Kruse, G. W. & Heiser, G. J. 2010**
*Computer Forensics Incident response essentials*, 14th printing, New York: Addison Wesley.

**Prosise, C. & Mandia, K. 2001**
*Incident Response: Investigating Computer Crime*, McGrawHill Osborne Media.

**Prosise, C. & Mandia, K. 2003**
*Incident response and computer forensics*, second edition, The McGraw-Hill Companies.

## REZIME

## DIGITALNA ARHEOLOGIJA "LAKO IZMENJIVIH" PODATAKA ZA LINUX OKRUZENJE

**Ključne reči**: digitalna arheologija, forenzička analiza, prikupljanja podataka, Linux sistem, prikupljanje "uživo", volatile data, kompromitovani sistem.

Prikupljanje, analiza i očuvanje podataka na "živom" sistemu jeste neophodnost da bi se blagovremeno utvrdilo da li postoji određena incidentna/protivpravna aktivnost. S obzirom na značaj prikupljanja podataka onih sa privremenim karakterom (eng. volatile data) u tekstu su opisani načini prikupljanja i alati koji se odnose na njihovo prikupljanje na Linux platformi. U radu su navedeni sledeći podaci privremenog karaktera koji se prikupljaju sa kompromitovanih sistema: sistemsko vreme i datum, postojeće mrežne konekcije, otvoreni TCP i UDP portovi, izvršni fajlovi koji otvaraju TCP i UDP portove, pokrenuti procesi i servisi, otvoreni fajlovi, interne tabele rutiranja i keš tabele, učitani moduli u kernel, sadržaj memorije i memorijskog procesa, montirani fajl sistema. U skladu sa navedenim ciljem ovaj rad je orijentisan na prikupljanje podataka iz "živog" Linux sistema sa tačno određenih mesta na sistemu. Na

tim mestima mogu se pronaći potencijalni dokazi koji mogu ukazati sa jedne strane na one forenzičke relevantne događaje koji utiču na bezbednost sistema, a sa druge mogu da potvrde ili ospore postojanje određene incidentne, odnosno protivpravne aktivnosti.