

HackMag (<https://hackmag.com>).

# Conducting Forensics on Linux: Memory, Disk, and Network Dumping for Evidence Collection

**Date:** 26/07/2025    **Author:** [Ivan Piskunov \(https://hackmag.com/author/g14vano\)](https://hackmag.com/author/g14vano)



The first task in digital forensics is information gathering, specifically acquiring images of hard drives and RAM, as well as, if helpful, network connection dumps. In this article, we will explore what needs to be done to obtain all of this on Linux machines and, at the same time, learn other useful skills.

This is a new installment in our forensic series for beginners, where we explain what digital forensics is, explore the most popular analysis tools, examine a few case studies on Android devices, and investigate the theft of funds from an online banking system on a Windows 10 laptop.

There are many ways to create a dump of a hard drive's contents or system memory. You can use native utilities ([https://en.wikipedia.org/wiki/GNU\\_Coreutils](https://en.wikipedia.org/wiki/GNU_Coreutils)) included in the distribution as well as third-party programs—available under both open-source licenses and commercial licenses. I will focus, where possible, on the most well-known, straightforward, and effective tools.

To begin with, for monitoring active systems, I recommend installing the utility Auditd. It provides detailed information on system changes in audit mode.

First and foremost, we will be focusing on events such as:

- System start and shutdown (reboot, halt)
- Reading/writing system files and modifying their access permissions
- Initiating network connections and changing network settings
- Modifying user or group information
- Changing date and time settings
- Installing, removing, starting, and stopping programs and daemons
- Executing system calls

It's also recommended to configure the system's `syslog.conf` and adjust the message collection rules: increase the depth of collected alerts (level 7) and expand the pool of sources (ranging from 0 to 15). This will allow you to monitor system changes in real-time.

## KEY ASPECTS OF FORENSICS ON LINUX

In a previous article focusing on a case of financial theft in online banking on Windows 10, we utilized programs with graphical interfaces whenever possible. Excluding proprietary solutions like EnCase Forensic (<https://www.guidancesoftware.com/encase-forensic>) or Belkasoft Evidence Center (<https://belkasoft.com/ec>), most tools on Linux operate in command-line mode.

The use of test commands can significantly save time when working with software; however, it might be too complex for beginners. But then again, beginners usually don't engage in forensic analysis! 😊

In addition to individual utilities, there are entire distributions designed for digital forensics. These include DEFT (<http://www.deftlinux.net/download/>), CAINE (<https://www.caine-live.net/>), Sumuri PALADIN (<https://sumuri.com/product-category/software/sumuri-software/paladin/>), Helix (<http://www.e-fense.com/>), and, of course, the well-known Kali Linux (<https://kali.tools/all/?category=forensic>).

When it comes to literature on Linux forensics, I would first recommend what is arguably the only comprehensive book on the topic: “Linux Forensics” by Philip Polstra. Second on the list is “UNIX and Linux Forensic Analysis DVD Toolkit” by Chris Pogue and others. Lastly, consider “Digital Forensics with Kali Linux.”

## **COMPREHENSIVE INSPECTION CHECKLIST**

To search for and gather forensic evidence, the first step will be to create images (dumps) of the following system objects:

- RAM (system and user processes, daemons, potentially running malicious code, etc.);
- Hard drive (sector-by-sector copy of the HDD, including deleted partitions, unpartitioned areas of the disk, shredded files, hidden files and directories, and more);
- Network stack (active connections, open ports, “unknown” services on ports, suspicious traffic).

Within the operating system itself, we will particularly focus on the following aspects:

- List of users, groups, and privileges.
- Processes running with root permissions.
- Scheduled tasks (cron jobs).
- Files with setuid and setgid bits.
- Contents of the `/etc/sudoers` file.
- Hidden files and directories.
- Files currently opened for reading.
- Network interfaces, connections, ports, and routing table.
- Logs from iptables and fail2ban (Reports, Alarms, Alerts).
- Configuration of `/etc/ssh/sshd_config`.
- Syslog daemon logs (checking for typical alerts).
- Status of SELinux.

- List of loaded kernel modules.

As an additional option, you can generate checksums for the main system files (for example, using the tool TripWire (<https://en.wikipedia.org/wiki/Tripwire>)) and later compare them with the standard values from the original distribution. However, this process is quite complex and requires a considerable amount of time and patience, so we will kindly skip the details.

Since the hardware running the entire operation is located in a secure data center, it rules out the need to search for artifacts related to Network File System (NFS) ([https://en.wikipedia.org/wiki/Network\\_File\\_System](https://en.wikipedia.org/wiki/Network_File_System)), locally mounted devices, and USB-connected devices.

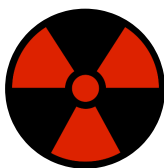


## **WARNING**

Always carefully consider the action you are taking and its purpose. Incorrect use of the programs mentioned in the article can lead to data loss (artifacts) or distortion of the acquired data (forensic evidence). Neither the author nor the editorial team is responsible for any damage caused by the improper use of the material in this article.

## **CREATING AN HDD IMAGE**

You can create a sector-by-sector copy of a hard drive without resorting to additional utilities. We'll use the classic and reliable native tool **dd**. This utility enables the creation of exact bit-by-bit copies of entire drives, specific partitions, or even individual files.

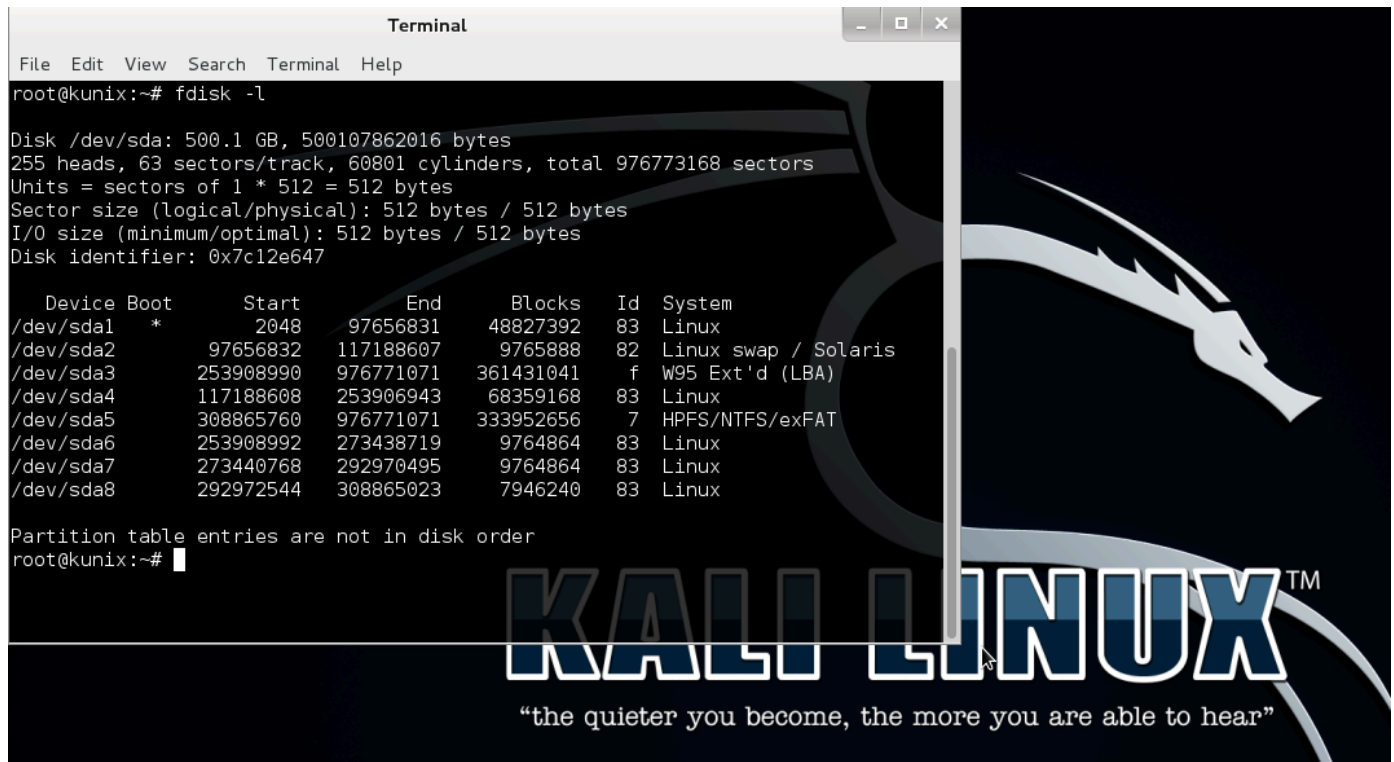


## **WARNING**

Using system tools for disk imaging and handling can lead to accidental writes, which is unacceptable in serious forensic investigations. Use the commands below only if you deem them appropriate for your situation.

First, let's retrieve a complete list of partitions from the system using the **fdisk** command:

```
$ fdisk -l
```



```
Terminal
File Edit View Search Terminal Help
root@kunix:~# fdisk -l

Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x7c12e647

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *         2048          97656831   48827392   83   Linux
/dev/sda2            97656832   117188607    9765888   82   Linux swap / Solaris
/dev/sda3        253908990   976771071   361431041    f    W95 Ext'd (LBA)
/dev/sda4        117188608   253906943    68359168   83   Linux
/dev/sda5        308865760   976771071   333952656    7    HPFS/NTFS/exFAT
/dev/sda6        253908992   273438719    9764864   83   Linux
/dev/sda7        273440768   292970495    9764864   83   Linux
/dev/sda8        292972544   308865023    7946240   83   Linux

Partition table entries are not in disk order
root@kunix:~#
```

Viewing the list of file systems

The basic syntax for the **dd** command is as follows:

```
$ dd if=<source> of=<destination> bs=<byte size>
```

For example, to create a copy of an HDD with a cluster size of 512 bytes:

```
$ dd if=/dev/sda1 of=/dev/sdb1 bs=512
```

During the process of copying an HDD, there may be damaged sectors. To ensure the software doesn't stumble on these and halt operations, you need to add the additional flag **-noerror**:

```
$ dd if=/dev/sda1 of=/dev/sdb1 bs=512 noerror
```

```
root@kali:~# dd if=/dev/sda5 of=/media/root/7CE2A8C8E2A887CA/Capture/swap.img conv=noerror,sync
6285312+0 records in
6285312+0 records out
3218079744 bytes (3.2 GB, 3.0 GiB) copied, 301.204 s, 10.7 MB/s
```

However, the best global practices involve using an enhanced version of the previous utility called **dcfldd** (<http://dcfldd.sourceforge.net/>). This tool was developed in the computer forensic laboratory of the DCFL

([https://en.wikipedia.org/wiki/Department\\_of\\_Defense\\_Cyber\\_Crime\\_Center](https://en.wikipedia.org/wiki/Department_of_Defense_Cyber_Crime_Center))

and includes several specialized options for creating dumps for forensic analysis. For instance, dcfldd can hash copied data and verify its integrity. It also displays the progress of the dump creation, logs the actions performed, and saves MD5 checksums in a separate file.

Example of command execution:

```
$ dcfldd if=/dev/sda1 hash=md5 of=/media/forensic_disk_image.dd bs=512
```

```
root@plr4te:~# dcfldd if=/dev/sdb hash=md5,sha1 md5log=/image/md5.txt sha1log=/i
image/sha1.txt of=/image/toshiba.dd
488192 blocks (15256Mb) written.
488320+0 records in
488320+0 records out
root@plr4te:~# cat /image/md5.txt
Total (md5): 21d95224f925977a6d170d6565399959
root@plr4te:~# cat /image/sha1.txt
Total (sha1): 6ec456084e6e65cf832ab4e9b6da58a142ce36be
root@plr4te:~# dcfldd if=/dev/sdb vf=/image/toshiba.dd verifylog=/image/verifyto
siba.txt
Total: Match
```

Output from the dcfldd utility

Instead of using the command line, you can opt for the graphical version of the **FTK** (<http://www.accessdata.com/forensictoolkit.html>) utility, which we are already familiar with from the previous article.

## CREATING A RAM DUMP

As with a hard drive, there are several ways to tackle this issue. Some options include:

- Use the native kernel module Linux Memory Extractor (<https://github.com/504ensiclabs/lime>) (**LiME**);
- Employ the script Linux Memory Grabber (<https://github.com/halpomeranz/lmg/>), which doesn't require installation and can be run from a USB stick, for example;

- Utilize the utility bundle lmap and pmem  
(<https://www.forensicswiki.org/wiki/Rekall>), which are part of the **Rekall** package.  
These are the tools I will be using.

A few words about Rekall (<http://www.rekall-forensic.com/>). It is a separate branch of the well-known Volatility Framework ([https://www.forensicswiki.org/wiki/Volatility\\_Framework](https://www.forensicswiki.org/wiki/Volatility_Framework)) written in Python, specifically designed for inclusion in forensic distributions that operate using Live CDs.

Navigate to the directory containing the tool and compile it from the source code:

```
$ cd rekall/tools/linux/  
$ make
```

We load the kernel driver `pmem.ko` into RAM:

```
$ sudo insmod pmem.ko
```

Checking the driver initialization:

```
$ sudo lsmod
```

After this, the driver creates a container file for our future RAM image: `/dev/pmem`.

Now, using the same dd utility, we create an image of the system's RAM:

```
$ dd if=/dev/pmem of=forensic_RAM_image.raw
```

Finally, we unload the driver after completing the task:

```
$ rmmod pmem
```

It's done!

# NETWORK TRAFFIC ANALYSIS

There are several tools commonly in use: primarily the command-line utility tcpdump (<https://en.wikipedia.org/wiki/Tcpdump>), the classic Wireshark (<https://www.wireshark.org/>), and the open-source framework XPLICO (<https://www.xplico.org/>), although the latter is more often utilized for subsequent data analysis rather than initial collection. For those who are not familiar with these programs, there are comprehensive tutorials available, such as using tcpdump for network traffic audits and maximizing the display filter capabilities of Wireshark.

Let's start with **tcpdump**. The basic command syntax looks as follows:

```
$ tcpdump <options> <filter>
```

Here are some of the most important options:

- `-i interface` – specifies the interface from which to analyze traffic;
- `-n` – disables IP address to hostname resolution;
- `-e` – enables display of link-layer data (e.g., MAC addresses);
- `-v` – provides additional information (such as TTL, IP options);
- `-w filename` – sets the filename to save the captured information (dump);
- `-r filename` – reads (loads) a dump from the specified file;
- `-q` – puts tcpdump into “quiet mode,” analyzing packets at the transport layer (protocols like TCP, UDP, ICMP) rather than the network layer (IP protocol).

We are capturing all incoming traffic from the internet to our server:

```
$ tcpdump -s 0 -i eth0 -n -nn -ttt dst host <IP address of our host> -v
```

Example of creating a network traffic dump for FTP or SSH protocols on the eth0 interface:

```
$ tcpdump -s 0 port ftp or ssh -i eth0 -w forensic_cap.pcap
```

We're dumping everything that goes through the eth0 interface.

```
$ tcpdump -w forensic_cap -i eth0
```



```

root@kali:~# tcpdump -i mon0
tcpdump: WARNING: mon0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on mon0, link-type IEEE802.11_RADIO (802.11 plus radiotap header), capture size 65535 bytes
00:17:33.246723 1039392549us tsft 1.0 Mb/s 2412 MHz 11b -32dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.322645 1039370165us tsft 1.0 Mb/s 2412 MHz 11b -30dB signal antenna 0 Data IV:3eb4 Pad 20 KeyID 2
00:17:33.349120 1039396645us tsft 1.0 Mb/s 2412 MHz 11b -29dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.453810 1039501335us tsft 1.0 Mb/s 2412 MHz 11b -27dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.553921 1039634213us tsft 1.0 Mb/s 2412 MHz 11b -25dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.656326 1039703845us tsft 1.0 Mb/s 2412 MHz 11b -24dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.758735 1039904549us tsft 1.0 Mb/s 2412 MHz 11b -23dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.861134 1039908645us tsft 1.0 Mb/s 2412 MHz 11b -22dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:33.963542 1040043813us tsft 1.0 Mb/s 2412 MHz 11b -21dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:34.065966 1040113445us tsft 1.0 Mb/s 2412 MHz 11b -20dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY
00:17:34.168364 1073737509us tsft 1.0 Mb/s 2412 MHz 11b -19dB signal antenna 0 Beacon () [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 1, PRIVACY

```

Output from tcpdump command

A suitable utility for our purposes is TCPflow

(<https://www.forensicswiki.org/wiki/Tcpflow>). Essentially, it's a more advanced version of tcpdump that supports additional filtering options and the ability to reconstruct "broken" packets.

If TCPflow isn't installed by default on your system, start by installing the `tcpflow` package.

The basic syntax of the command is as follows:

```
$ tcpflow [options] [expression] [host]
```

Here is a description of the options:

- `-c` – console output only (do not create files);
- `-d` – debugging level (default is 1);
- `-e` – display each stream in alternating colors (blue for client-server, red for server-client, green for unknown);
- `-i` – network interface for listening;
- `-r` – read packets from a tcpdump output file;
- `-s` – remove non-printable characters (they will be replaced with dots).

An example of data collection from an external network to our server:

```
$ tcpflow -ce host <IP address of our host>
```

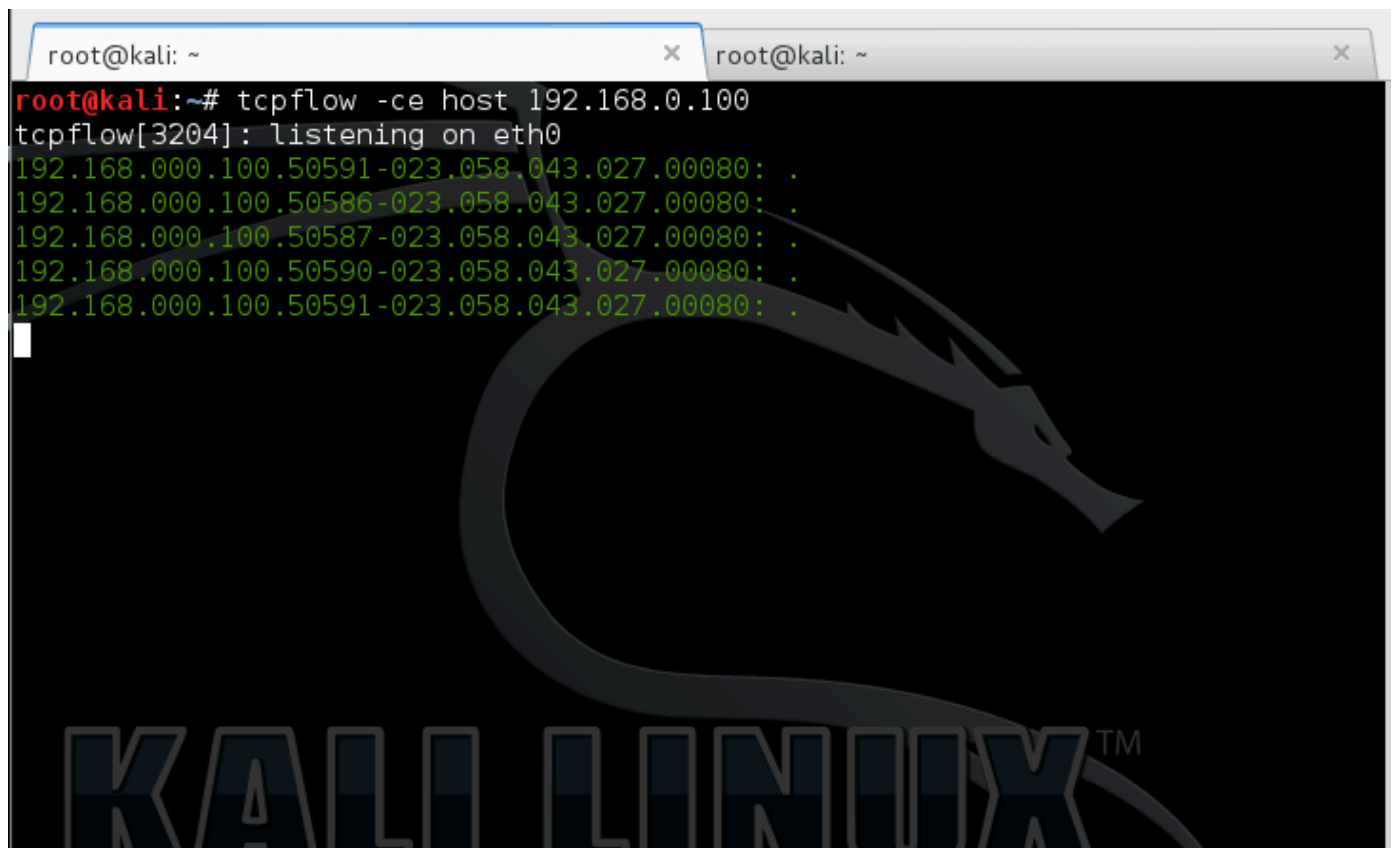
Capturing all HTTP traffic on our network:

```
$ tcpflow -ce port 80
```

Dumping network traffic data to a local folder:

```
$ mkdir tcpflowdata
$ cd tcpflowdata
$ tcpflow host <IP address of the target machine>
```

Files containing the content of network connections will now be stored in the `/tcpflowdata` directory. All we need to do afterwards is to transfer them to a parser for analysis.

A screenshot of a terminal window on a Kali Linux system. The window has two tabs, both labeled 'root@kali: ~'. The terminal shows the command 'root@kali:~# tcpflow -ce host 192.168.0.100' being executed. Below the command, it says 'tcpflow[3204]: listening on eth0'. Then, five lines of green text show network traffic: '192.168.0.0.100.50591-023.058.043.027.00080: .', '192.168.0.0.100.50586-023.058.043.027.00080: .', '192.168.0.0.100.50587-023.058.043.027.00080: .', '192.168.0.0.100.50590-023.058.043.027.00080: .', and '192.168.0.0.100.50591-023.058.043.027.00080: .'. The background of the terminal window features a large, stylized dragon logo and the text 'KALI LINUX™'.

Output from TCPflow

## ACCESSING COLLECTED DATA

The disk and memory images are ready. Now we need to mount them on the research machine to start searching for and analyzing artifacts. Mounting a HDD image is straightforward; the output file generated by `dd` or `dcfldd` can be easily mounted as a new device.

The syntax for the `mount` command is as follows:

```
$ mount -o,ro,noatime,loop,offset= <offset from the beginning of the pa
```

For instance, to mount a previously created HDD dump, you need to execute the command

```
$ mount -o,ro,loop forensic_disk_image.dd /mnt/temp/
```

We will use Volatility Framework (<https://code.google.com/archive/p/volatility/>) to analyze the RAM image. More details will be provided in the section “Artifact Hunting” below.

A .pcap file containing network connection dumps can be opened in Wireshark or XPLICO. However, I prefer using services with semi-automatic parsing, such as PacketTotal (<https://packettotal.com/>), Pcap Analyzer (<https://pcap.honeynet.org.my/v1/>), or the paid CloudShark (<https://cloudshark.io/>).

## ADDITIONAL USEFUL COMMANDS

Here are a few commands that will help us gather anything we might have missed. Some of these are just in case, as they won’t take much time.

### Instantly Capture System State and Key Configuration Details

```
#!/bin/bash
dmesg > dmesg.txt
cat /proc/mounts > proc_mounts.txt

# Collect checksums from all currently mounted devices
for p in $(md5sum /proc/mounts /proc/*/mounts | sort | uniq -d -w 32 |
cat $p > ${p////_});
done

cat /proc/mdstat > proc_mdstat.txt
lspci > lspci.txt
uname -a > uname_a.txt
uptime > uptime.txt
```

### Remount All File Systems as Read-Only

For each file system we will be examining, we’ll remount it in `ro` (read-only) mode:

```
// set mountpoint  
  
MOUNTPOINT=/home  
mount -o remount,ro ${MOUNTPOINT}
```

Let's create a timestamp or timeline:

```
find "${MOUNTPOINT}" -xdev -print0 | xargs -0 stat -c "%Y %X %Z %A %U %
```

## Obtaining the Status of Existing Connections and Open Sockets

```
// --verbose --wide --extend --timers --program --numeric (--listening)  
  
netstat -v -W -e -o -p -n > netstat_vWeopn.txt  
netstat -v -W -e -o -p -n -l > netstat_vWeopnl.txt  
  
// same without --numeric  
netstat -v -W -e -o -p > netstat_vWeop.txt  
netstat -v -W -e -o -p -l > netstat_vWeop1.txt
```

## Current ARP Cache Dump

```
arp -n > arp_n.txt  
ip neigh show > ip_neigh_show.txt
```

## Capturing the Current State of iptables Network Filter Rules

```
// --verbose --numeric --exact --list --table  
for table in filter nat mangle raw; do iptables -v -n -x -L -t ${table};  
for table in filter mangle raw; do ip6tables -n -t ${table} -L -v -x >  
for table in filter nat broute; do ebtables -t ${table} -L --Lmac2 --Lc
```

## Dumping ipsets (commonly used by fail2ban and firewallld)

```
ipset list > ipset_list.txt
```

## Saving the Process Table

```
ps auxwww > ps_auxwww.txt
```

Another option for more readable output:

```
pstree -a -l -p -u > pstree_alpu.txt
pstree -a -l -p -u -Z > pstree_alpuZ.txt
```

## Preserve the Original Location of Executable by PID

```
ls -l /proc/${PID}/ > proc_${PID}_ls_l.txt
cat /proc/${PID}/exe > proc_${PID}_exe
```

## Viewing All Open Files on the System

If a file has been deleted, the `ls` command will append the flag `(deleted)` to the destination file's name. However, the content can still be accessed using symbolic links in `/proc/${PID}/fd`. This is often helpful when dealing with malware written in interpreted languages like Perl and Python. Therefore, for further analysis, we will save all open files:

```
ls -l /proc/${PID}/fd > proc_${PID}_fd.txt

// copy interesting open files, substitute MYFD with file descriptor number
MYFD=1234
cat /proc/${PID}/${MYFD}> proc_${PID}_fd_${MYFD}
```

## CONCLUSION

Now we have everything needed for further analysis. In the next and final article of the series, I will demonstrate how to use the gathered data to identify artifacts. To make it more engaging, I have an interesting case study involving the detection of malware that infiltrated a hosting provider's servers. Stay tuned!

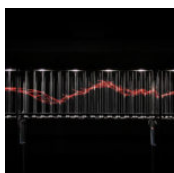
## Related posts:



2022.06.03 — **Vulnerable Java. Hacking Java bytecode encryption**  
(<https://hackmag.com/security/java-bytecode-encryption>)

Java code is not as simple as it seems. At first glance, hacking a Java app looks like an easy task due to a large number of available...

**Full article →** (<https://hackmag.com/security/java-bytecode-encryption>)

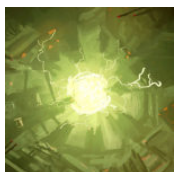


2022.02.15 — **Reverse shell of 237 bytes. How to reduce the executable file using Linux hacks** (<https://hackmag.com/coding/reverse-shell-237-bytes>)

Once I was asked: is it possible to write a reverse shell some 200 bytes in size?

This shell should perform the following functions: change its name...

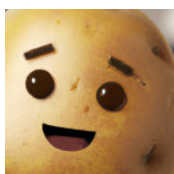
**Full article →** (<https://hackmag.com/coding/reverse-shell-237-bytes>)



2022.02.09 — **Kernel exploitation for newbies: from compilation to privilege escalation** (<https://hackmag.com/coding/linux-kernel-exploitation>)

Theory is nothing without practice. Today, I will explain the nature of Linux kernel vulnerabilities and will show how to exploit them. Get ready for an exciting journey:...

**Full article →** (<https://hackmag.com/coding/linux-kernel-exploitation>)



2023.03.26 — **Poisonous spuds. Privilege escalation in AD with RemotePotato0** (<https://hackmag.com/security/remotepotato0>)

This article discusses different variations of the NTLM Relay cross-protocol attack delivered using the RemotePotato0 exploit. In addition, you will learn how to hide the signature of an...

**Full article →** (<https://hackmag.com/security/remotepotato0>)



2022.06.02 — **Climb the heap! Exploiting heap allocation problems** (<https://hackmag.com/security/heap-allocation>)

Some vulnerabilities originate from errors in the management of memory allocated on a heap. Exploitation of such weak spots is more complicated compared to 'regular' stack overflow; so,...

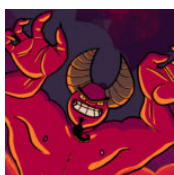
**Full article →** (<https://hackmag.com/security/heap-allocation>)



2022.02.09 — **Dangerous developments: An overview of vulnerabilities in coding services** (<https://hackmag.com/devops/coding-services-bugs>)

Development and workflow management tools represent an entire class of programs whose vulnerabilities and misconfigs can turn into a real trouble for a company using such software. For...

**Full article →** (<https://hackmag.com/devops/coding-services-bugs>)



2022.06.01 — **Log4HELL! Everything you must know about Log4Shell** (<https://hackmag.com/security/log4hell>)

Up until recently, just a few people (aside from specialists) were aware of the Log4j logging utility. However, a vulnerability found in this library attracted to it...

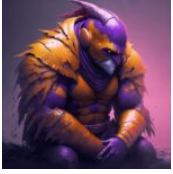
**Full article →** (<https://hackmag.com/security/log4hell>)



2023.07.07 — **VERY bad flash drive. BadUSB attack in detail**  
(<https://hackmag.com/security/very-bad-usb>).

BadUSB attacks are efficient and deadly. This article explains how to deliver such an attack, describes in detail the preparation of a malicious flash drive required for it,...

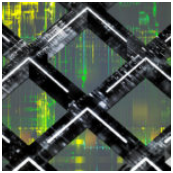
**Full article → (<https://hackmag.com/security/very-bad-usb>)**



2023.04.20 — **Sad Guard. Identifying and exploiting vulnerability in AdGuard driver for Windows** (<https://hackmag.com/security/aguard-cve>).

Last year, I discovered a binary bug in the AdGuard driver. Its ID in the National Vulnerability Database is CVE-2022-45770. I was disassembling the ad blocker and found...

**Full article → (<https://hackmag.com/security/aguard-cve>)**



2022.01.12 — **Post-quantum VPN. Understanding quantum computers and installing OpenVPN to protect them against future threats**  
(<https://hackmag.com/security/quantum-vpn>).

Quantum computers have been widely discussed since the 1980s. Even though very few people have dealt with them by now, such devices steadily...

**Full article → (<https://hackmag.com/security/quantum-vpn>)**