Forensic analysis

Laboratory for the class "Cybersecurity" (01UDR)
Politecnico di Torino – AA 2021/22
Prof. Antonio Lioy

prepared by:
Diana Berbecaru (diana.berbecaru@polito.it)
Andrea Atzeni (andrea.atzeni@polito.it)

v. 1.1 (07/12/2021)

Contents

1 Purpose of this laboratory

In this laboratory, you will perform exercises aimed to experiment with forensic analysis, specifically targeting analysis of the storage.

The opinion that deleting a file or history through a command like `rm` or dropping it in the Trash will remove it completely from the hard disk drive is wrong. In reality, it only removes the file from the logical structures that address it, but the actual file remains on your computer until overwritten due to (future) requests of storage space from the OS and its applications.

Additional software tools

The tools listed below will be used as well throughout this laboratory:

dd - 'dd' copies a file (from standard input to standard output, by default) to/from an I/O block size, while optionally performing conversions on it.

Warning: Some people believe dd means "Destroy Disk" or "Delete Data" because if it is misused, a partition or output file can be trashed very quickly. Since `dd` is the tool used to write disk headers, boot records, and similar system data areas, misuse of `dd` has already caused enough pain...

dcfldd - 'dcfldd' is an enhanced version of GNU dd with features useful for forensics and security. Based on the dd program found in the GNU Coreutils package. It has some additional features particularly suited for forensics analysis, like *hashing-on-the-fly*, flexible data-wipes and cloning verification
Home page = `https://dcfldd.sourceforge.net/`

exiftool - ExifTool is a platform-independent Perl library plus a command-line application for reading, writing and editing meta information. Reads EXIF, GPS, IPTC, XMP, JFIF, MakerNotes, GeoTIFF, ICC Profile, Photoshop IRB, FlashPix, AFCP, ID3, Lyrics3, ODT, DOCX. For a complete list of supported formats see `https://exiftool.org/#supported`
Home page = `https://exiftool.org`

foremost - Foremost is a console program to recover files based on their headers, footers, and internal data structures. This process is commonly referred to as data carving. Foremost can work on image files, such as those generated by dd, Encase, etc, or directly on a drive
Home page = `http://foremost.sourceforge.net/`

photorec - PhotoRec is file data recovery software designed to recover lost files including video, documents and archives from Hard Disks and CDRom and lost pictures (Photo Recovery) from digital camera memory. PhotoRec ignores the filesystem and goes after the underlying data, so it'll work even if your media's filesystem is severely damaged or formatted
Home page= `www.cgsecurity.org`

First of all, for performing a forensics analysis, you have to create a trusted environment in which to perform your analysis without be afraid of compromisison. You can choose two different way of peroming, both of them valid from a digital forensic perspective: 1) run a live OS and move you evidences in there, or 2) run a Virtual machine and insert the evidences inside the virtual environment (in this case, both host and guest OS must be trusted).

In the first case, you can run Kali linux in the laboratory as usual.

If you want to experiment the second case, you can download CAINE (sec. 2) (we provided the last version of the image inside Polito premises, that should allow a quick download from Labinf), run VirtualBox and use the iso image to configure the virtual machine.

2 CAINE

CAINE (Computer Aided INvestigative Environment) is an Italian GNU/Linux live distribution created as a Digital Forensics project in 2008, originally developed at the University of Modena and Reggio Emilia, specifically suited for Computer Forensics analysis (detailed history can be seen at `https://www.caine-live.net/page4/history.html`)

As a distro, it integrates software tools as modules along with powerful scripts in a GUI. In figure. 1 you can see the appearance of the running live.

The first operation to perform the laboratory is to download the ISO image of the last version; a mirrored image can be found here:

`https://storage-sec.polito.it/external/caine/2020/caine11.0.iso`

Since the integrity and the accuracy of working tool is of the uttermost importance in a forensics analysis, the first thing to do is to check the status of the instruments used. Different hash computation of the ISO image are the following:

Figure 1: CAINE v11

```
MD5 - 73EA6E4F3B1861EFC1472B891DFA1255 caine11.0.iso
SHA1 - 74E059AF4547CB5D765080BDB8B236E4CB4550AE caine11.0.iso
SHA256 - 30a3cdf4012f08317eacc562f1b1b120e39ea5fda6c8772a95e32f3be8183d0c  caine11.0.iso
```

Virtual Box

After that, you can run Live CAINE from an ISO file by mounting it on the virtual DVD of an ad hoc VM. Follow the instructions we have provided in the "general laboratory instructions" in the text of the first laboratory, in the section 2.1.2. The only change is that you have to use `caine11.0.iso` as DVD image.

Note: experiments have shown unexpected malfunction if the VM has low RAM capacity. The advice is to choose 4 GB or more, and in any case no less than 2 GB.

Adding a Virtual disk

After the creation of a VM, in order to make available another device to attach other images, you need to define a new empty virtual disk in the virtual box set-up. To this aim, you need to perform the following steps:

1. select the CAINE VM (before starting it)
2. select the icon `Settings` and then select `Storage`
3. add a new `IDE` virtual hard disk by clicking on the `Adds Hard Disk` file on the right of `Controller:IDE` row
4. `Create` a new `VirtualBox Disk Image (VDI)` choosing 5 GB as HD size, and then `Choose` it in order to attach to the current VM

In this way, inside CAINE another device corresponding to the virtual one will be present.

3 Image acquisition

Status of the Virtual Environment

In a Computer Forensics analysis, it is very important to avoid accidental modification of the evidence and data under analysis. To avoid data modification, almost all Linux-based recent distribution suitable for Computer Forensic uses a default read-only mounting of the present devices (this is true for CAINE and for Kali in `forensic mode`).

Take awareness of the present initial condition your system:

- Linux environments allow multiple possibilities to investigate and manage file system structure. From a command-line tool perspective, a direct way to acquire info on devices and partition is the `fdisk` command-line tool, and information on its usage can be found through the command `man fdisk`.

  Write the syntax to see the available device(s) through `fdisk`.

  →

  List all available virtual devices in the system, and identify the virtual device suitable for your investigation.

  →

  Identify if they are in read-write or read-only state. Pay attention that the question is not if they are *mounted* in read-only or read-write mode (that could be investigated by using the `mount` command), but instead if the `device` is in read-only or read-write state. To investigate that, you can use the `blockdev` command:

  ```
  blockdev --report devicename
  ```

  and check the result of the first column (`ro` means read-only, `rw` means writable).

  →

Acquisition of the image

A very common task in a Computer Forensics analysis is the acquisition of part of a storage/hard disk in order to perform deep analysis. In Linux environment two main possible option exists: 1) to acquire a physical storage, physically integrate it, and see it as a device to connect to the file system (e.g. by mounting them),or 2) the original storage can be deposited in a single file that contains the whole content and file system structure inside (using tools like `dd` and the forensics counterparts).

In case a file containing a valid file system is available, a possibility to investigate it is by using a functionality to mount these images files as valid block devices. Many Forensics Linux distribution provides some GUI interfaces to do that (like `Disk Image Mounter` for CAINE). Those commands use the `loop` interface for devices. Loop devices are artefacts that allow access to a file in the same way of accessing to a block device. This is achieved by introducing a pseudo-block device (the `loop` one) as intermediary to access to the original file. At OS level, the operations (`read`, `write`, . . .) to the mounted directory will be re-directed versus the loop device, which will transparently translate the requests into operation to the original file. When the `loop` device is already enabled (like in CAINE and Kali) it can be directly used through the `mount` command:

```
mount -o loop image.dd mounting_directory
```

where *mounting_directory* must be an existent directory (e.g. `/media/imagedd`). Note that if you run `fdisk -l` before and after the `mount` command, you may appreciate the presence of a new `/dev/loop` device, which is the interface between `image.dd` and the *mounting_directory*.

Now, let's acquire our image: download it at the following URL:

https://storage-sec.polito.it/external/caine/2020/image2021.dd

and then check if the download did not introduce errors (or even if a malicious user manipulated it) by comparing the result of the download with the following digests:

```
MD5 - b80cdd38f67761b4bf00a75838cd4745  image2021.dd
SHA1 - e8336744d56e7c8627ae11bcb1d40add43308001  image2021.dd
SHA256 - 048aa27d608ab6e6b342e947ef86d1a28c7d9781d1e28f6d4d43d99ddff7bb22  image2021.dd
```

If everything is fine, a possible next step is to clone the image on a virtual device to further examine it.

To do so, you can use the `data dump (dd)` command indicating as source the image file, and as destination your virtual device.

```
dd if=image.dd of=/dev/virtual_device
```

Where the *virtual_device* is the device defined in the virtual box setup and identified in a previous step (e.g. `/dev/sda`). After the above command, a useful check again is checking that no errors have been introduced. A first rough check is that the number of input bytes is the same of the number of output bytes (these messages are provided by `dd` itself)

To check the integrity, do you think the above control is enough?

→

If you think it is not enough, what commands do you suggest to use in addition?

→

Alternatively, you might use the `dcfldd` tool that can automate the copy and the hash evaluation in a single step:

```
dcfldd if=image.dd of=/dev/virtual_device hash=md5,sha1,sha256 md5log=image.md5
sha1log=image.sha1 sha256log=image.sha256
```

where the *virtual_device* is the device defined in the virtual box setup and which has been identified in a previous step.

Now try to mount the freshly created device. CAINE inserts in the `/etc/fstab` an entry that maps devices in the `/dev` to a corresponding directory in `/media` (e.g. `/dev/sda1` is mapped on `/media/sda1`), so you can just use a command like:

```
mount /media/virtual_device
```

If everything went well, you can move inside that directory and explore the content of the disk.

Do you think that the "cloning" process performed with `dd` on `image.dd` can be done with every kind of image?

→

hint: by using the `file` command on the image `image.dd` you can have some clues

What is the file system of `image.dd`?

→

NOTE: in principle, you can create an image file from a virtual device using the same command, just reverting input and output

```
dd of=image.dd if=/dev/virtual_device
```

However, in this case you should know the size of the image and use it as parameter in the command. For example, you can use `findmnt` command-line tool to query the mounted device (in particular, with `findmnt -bno size` *mount_dir* you will find out also the size of the device), check the sector size of the device (e.g. with `fdisk -l` taking note of the sector size of the device) and then use the resulting knowledge to instruct the `dd` command, for example:

```
dd if=virtual_device, of=image.dd bs=sector_size count=number_of_sectors
```

4 File identification

Meta-data analysis

Now that your environment is ready and you have acquired the data to be analysed, it is time to start the analysis phase. Go into the *mounted_directory* and have a look at the content.

By using specific commands to analyse meta-data like `exiftool` walk through all the files and perform the following tasks:

- identify the file type;
- annotate interesting data about each file (e.g. creation date, last modification);
- note down any suspect detail that you encountered.

→

Are you able to delimit/individuate the time frame of activity for the device?

→

In the assumption this image is the exact content of a seized Hard Disk image, in your opinion does the CF agents that seized the device made any mistake?

→

File signature

Most of the known file formats can be identified by the few starting (and sometimes ending) bytes

A comprehensive list of file signature is publicly available at https://en.wikipedia.org/wiki/List_of_file_signatures.

Open the `.mp4` file with and hex viewer (e.g. `hexdump`, `hexedit`, `ghex`) and compare the first three bytes with the file signatures in the file signature map. It confirms your beliefs?

→

The command line utility `file` automatically performs the check against the signature, so it can be used to perform the check that has been performed at hand in the previous point thanks to the hex viewer.

Check all the files under exam with `file` and annotate if the identification at the previous point is confirmed or changed.

→

Finally, if you did not annotate this in previous points, try to determine the program that created each file. Optionally, also try to determine if any manipulation has been done on the different files present on the Hard Disk. Another helpful command in this operation is the `strings` command line tool, which allow to find out the different strings in a target file. Often, this allows for further information in a not very well know possible source of data.

→

By the way, while you are trying to discover creator of `very.mysterious`, have you noticed anything strange?

→

5 File carving

At this point you should have a pretty good knowledge of what were the files, and possibly the file content, of the visible information contained in the image . . . but what about the *invisible* content (i.e. deleted files)?

Deleted files can be a very rich source of information, but to recover them the file system structure have to be partially or totally bypassed (e.g. do not rely on allocation table). CAINE provides tools that do not rely on the file system structures to identify information, like `foremost`.

The process of extracting data from a device or disk image is named `carving`, meaning that the data comes out from a non-shaped image like sculptures from the stone (in this case the "sculptures" assume the form of, for example, .jpeg, .png, .zip, .pdf, files). We will instruct `foremost` to carve how many files as it can with the following command:

```
foremost -t all -i virtual_device -o recovery_directory -v
```

.

- `-t` set the file types to carve out, `all` means any supported file by foremost
- `-i` is specifying the input, which can be an image file or a device
- `-o` is the output directory, where the carved result will be placed. Inside the recovery directory foremost will create a set of directory for each file type will be able to recover
- `-v` log all the messages to an audit report (inside the recovery directory) instead of standard output)

What is the output? Has foremost been able to bring up new information?

→

Another powerful tool for data carving present in many CF distribution (like CAINE and Kali) is `photorec`. To execute it at best, some previous knowledge on the target of examination is expected, e.g. the file system type. Since you should alredy posses all required info from previous steps of this lab, you can execute the following command:

```
photorec virtual_device
```

And provide the required info, restricting the research to the unallocated space

Compare the result with foremost ones. Do you notice any difference?

→

6 Data wiping

To begin a new investigation, the testbed should be left perfectly clean to avoid pollution of the evidences of the next case.

To remove any trace of the evidence on the virtual device used so far, the command `dcfldd` can be used again. In this case, to perform data wiping. It allows overwriting every byte of the target device with a specific pattern. A sequence of zeros will perfectly accomplish the goal:

```
dcfldd pattern=00 of=virtual_device
```

After this operation, are you still able to mount the device?

→

Are you still able to identify the file system?

→

Try to perform some recovery task (e.g. `foremost` or `photorec`) and/or analyze the content with an hex viewer. Is there any sign of the previous data?

→

7 Additional exercises (optional)

Trying to hide the traces...

Exploiting the knowledge acquired in the previous exercises, prepare an image of 768 MB, and perform some file operations (e.g. creation and modification). Then, try to obfuscate the information related to those operations. For example, you might create and manipulate a picture, and then modify the picture details and metadata (e.g. creation time)
The image must accomplish the following requirements

- must have a well-known file system
- must contain one or more files
- must have a size that makes possible save them in a set of USB pen drives of 256 MB

The purpose of this exercise is to create, manipulate and alter the files and their appearance to make them complex to be identified, exploiting the tools used in the previous sections of this lab.
Then, you can transfer the image to your classmate and, she/he has to analyse the image and try to find out

- the file system type
- the number of files (including the cancelled ones)
- the number of attempts to conceal/manipulate the original information related to the files
- the details of the attempts (e.g. “the file extension has been changed from ”.mov“ to ”mp4“)

And at the same time, you can get an image built by your classmate, and perform the same task

$\rightarrow$

Finally, compare your result with the one of your classmates, and see who found most clues :)