INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

FORENSICS CYBER-SECURITY

MEIC, METI

Tutorial II

Introduction to Cyberforensic Investigations

2019/2020

nuno.m.santos@tecnico.ulisboa.pt

Introduction

This guide aims to provide a hands-on introduction to cyberforensic investigations. Since time constraints prevent us from covering all stages of the forensic process in detail, i.e., from assessment down to reporting stages, this tutorial will focus primarily in giving you some basic training in stages that require more creativity and insight from the digital investigator, namely collecting and analyzing electronic evidence. The most important outcome from this exercise should be the understanding of the forensic process leading from capturing evidence in unaltered form, through the examination of suspicious activities. During all steps the forensic principles must be followed, keeping impact on data as low as possible, saving the state of data before altering it and documenting all the alterations, tools and methods used.

We present you an exercise in which you can practice such skills in the context of a bank fraud case involving phishing. Let's assume that, during a phishing case investigation, one of the bank's customers identified as hit by phishing malware agreed to cooperate with the bank's CSIRT[1] team, which assigned you the job of conducting an investigation. You arrange a meeting during which you will be able to examine his personal computer for traces of malicious software. Your goal is to find evidence of malware activity in the customer's computer. For simplicity reasons, we assume that you have already cold cloned the hard disk and preserved the respective image on a forensic station. This exercise has two parts: gather live data from a working system, i.e., his personal computer (Section 2), and check obtained data for forensic value (Section 3). Throughout this exercise, create a document and a folder for keeping track of all your steps and collected evidence, respectively.

1 Preparing the environment

Before we begin, we must prepare the environment for forensic analysis. We provide you with a virtual machine (VM) package which has been prepared with the tools you need and digital artifacts to be inspected. This section explains how to set up this VM on your workstation. Start by downloading the VM package from: `http://turbina.gsd.inesc-id.pt/csf1920/enisa-main.ova`. Once you download this OVA package, import it into VirtualBox and instantiate a VM as described in the following steps. Note Virtualbox must be installed on your workstation.

1.1 Import the OVA package

To import the OVA package, open the "Oracle VM VirtualBox Manager". In the File menu, select "Import Appliance". The "Import Virtual Appliance" wizard is displayed in a new window. Select the downloaded OVA package and click "Next".

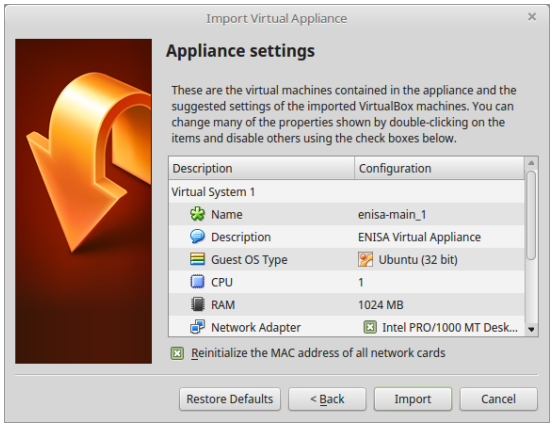

Figure 1: Appliance settings.

[1] CSIRT, Computer Security Incident Response Team

You can make adjustments to the default settings, as displayed in Figure 1, (you can change settings later if needed). It is recommend that you mark the "Reinitialize the MAC address of all network cards" checkbox. Next, click the "Import" button. After successfully importing the appliance, the wizard will close and the imported virtual machine will be listed in Oracle VM VirtualBox Manager.

This task can also be executed in command line (for the stations at Taguspark campus' laboratories) as explained below:

```
# make an attempt without changing the system and
# list the options to import
$ VBoxManage import enisa-main.ova --dryrun
# import the image
$ VBoxManage import enisa-main.ova --vsys 0 --vmname enisa-main
```

If you are following this guide in the stations at Alameda's RNL labs, you must first convert the OVA package to a suitable format and import it as explained below:

```
# extract the files within the OVA package
$ tar xvf enisa-main.ova -C /tmp
# convert the VMDK image to qcow2
$ qemu-img convert -O qcow2 /tmp/enisa-main-disk1.vmdk /tmp/ENISA.qcow2
# start the QEMU/KVM virtualization manager
$ rnl-virt manager
```

Then, choose *QEMU/KVM User Session* → *New*. Select Import and navigate to `/tmp` to import the generated `.qcow2` disk and configure the VM.

1.2 Boot the VM

Select the virtual machine you have imported and click on the "Start" button to start it up. Or start up the imported virtual machine using command line as follows:

```
$ VBoxManage startvm enisa-main
```

On the first run of the virtual machine, you will be asked to enter username and password. The credentials for the virtual machine are as the following (case sensitive):

- Username: enisa
- Password: enisa

After you have logged onto the virtual machine you will need to run an initial setup to initialize the exercise environment. This should only be done on the first run of the virtual machine. To do so open up "LXTerminal" by clicking the start menu in the bottom left corner and select "Accessories" -> "LXTerminal" as shown in Figure 2.

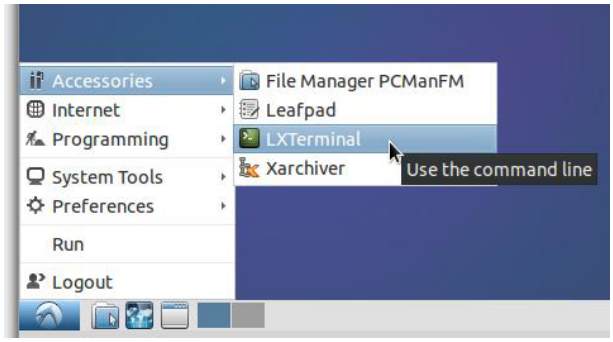

Figure 2: LXTerminal selection.

If you have a keyboard with Portuguese layout, you can adjust the virtual machine keyboard as follows:

```
enisa@10:~\$ setxkbmap pt
```

Now, configure the exercise environment in the LXTerminal window, typing the following command and press "Enter" to continue:

```
enisa@10:~\$ ./setup.sh
```

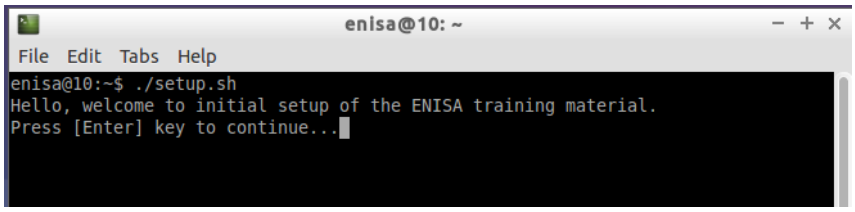

Figure 3: Run setup.

The complete setup will take approximately five minutes depending on the processing power of your host machine. When the setup phase is complete, press "Enter" again. This will force the system to reboot as shown in Figure 4.

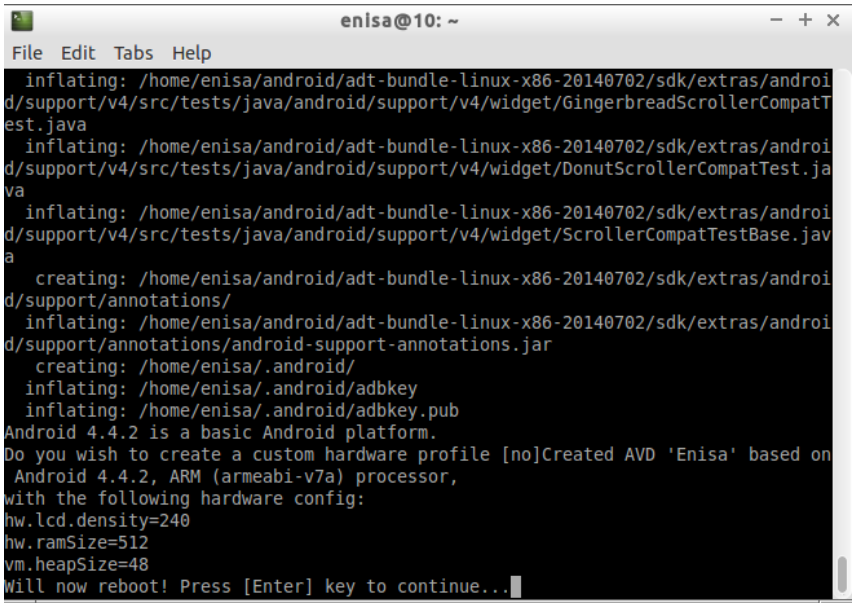

Figure 4: Finish setup.

After the reboot, all the required material will be displayed on the desktop under the folder "TrainingMaterial". You can now continue with the specific exercise instructions, and enjoy the session.

2 Gather live data

As we now have a full system image stored safely for further examination (or for restoring the computer to its original state) we can move on to the next step. In this step we will try to gather some evidence without installing or running any external software on the computer. While it must be stressed, that powering on the computer and running any software on it will change the system state, some evidence must be gathered in the runtime for behavioural analysis. To make the case acceptable (remember, we've secured the original state) we must mimic a regular users behaviour – by performing typical actions the

user would perform on a day to day basis. We make the least possible impact, trying to see if there are any anomalies in system's behaviour. In the very nature of malware we may expect in this case that there is persistence. In all cases but some very specialised and targeted attacks, malware installs a persistent part and tries to hide and stay in the system forever. To make it even worse – there is known malware that activates only when user activity is detected, mouse clicks in case of Trojan.APT.BaneChant for instance. There is virtually no way of capturing malicious activity in that case without simulating a regular user.

As we cannot provide a copy of an infected Windows operating system, we've prepared a PCAP file with traffic from the actual examined computer. The file name is `network.pcap` and it is located in folder `/home/enisa/Desktop/Training-Material/Dig_For_and_Handl_Ev`. We did not use the computer during the network capture in this case, all the traffic logged comes from system components or from installed software communicating without user interaction, possibly malicious software. Load the file to Wireshark:

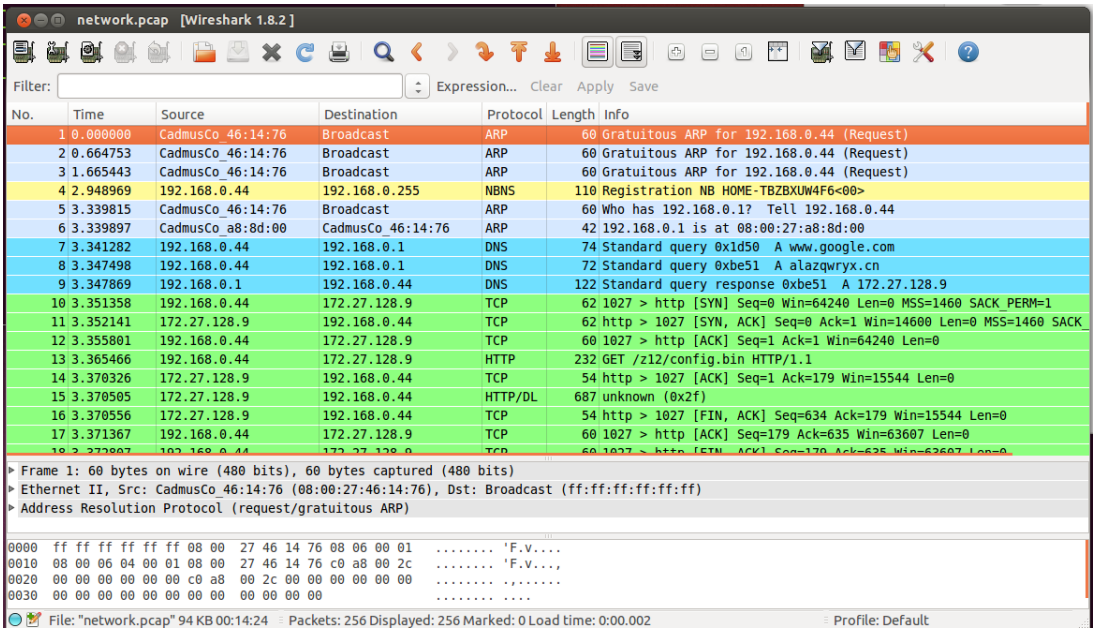

Figure 5: Network traffic loaded into Wireshark.

This capture contains data from the moments just after the computer was booted. We can see some network activity, so the computer tried to communicate with the external world. After a few ARP packets (the protocol that performs IP address to MAC address translation) some DNS queries were made. We decide to filter the traffic to DNS only to see what addresses the computer tries to find automatically. We type 'dns' in the 'Filter' field and accept the input by pressing ENTER:

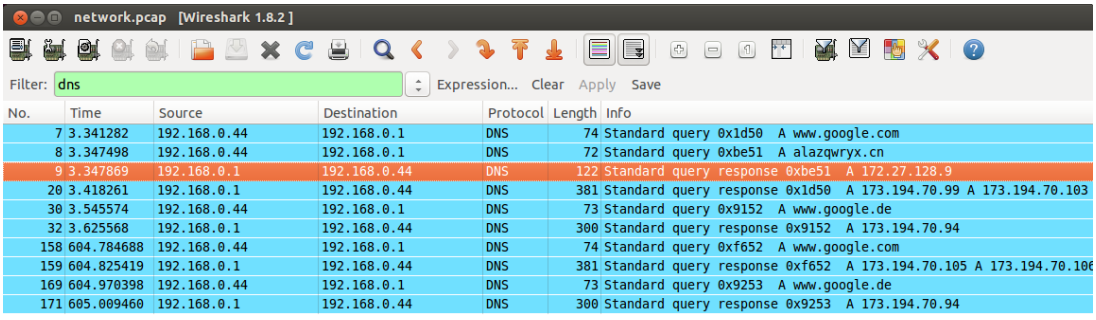

Figure 6: Possible malware trace in network traffic.

There are Google servers prevailing in the log, but one query catches our attention: The computer queried about the IP address of 'alazqwryx.cn' and got a response with IP address '172.27.128.9', high-

lighted in this picture. This is one of common patterns used by malware – asking for 'unidentified addresses' which are the most probably botnet CC servers. CC servers are contacted by malware for configuration or to send harvested data. Actually, we can see in previous picture, that the computer was contacting this IP address over HTTP protocol. We modify the filter to see what the communications looked like:

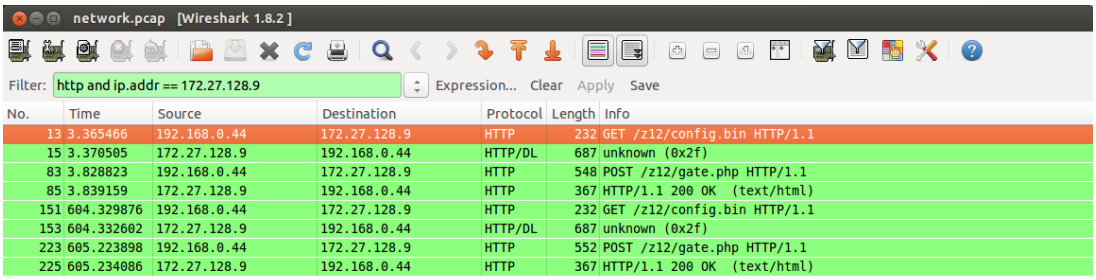

Figure 7: Suspected malware traffic filtered out.

New filter shows only the HTTP protocol where the IP address in question is used. We notice two HTTP requests:

```
GET /z12/config.bin
POST /z12/gate.php
```

If we look in the Time column we see that the pattern repeated after roughly 600 seconds (10 minutes). All this information combined suggests that we have a computer infected with malware and the HTTP requests format suggests this is some ZeuS variant. We have also acquired probable botnet's CC server IP address. Now that we know what we're looking for, we have to gather more traces. First, we make a dump of physical memory inside the workstation:

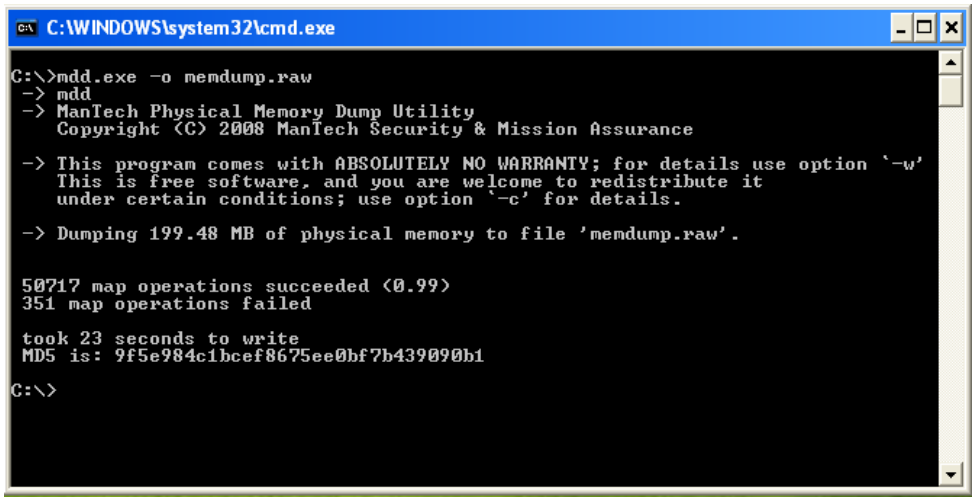

Figure 8: Acquiring memory.

We use a small and simple tool called mdd6[2] This tiny utility copies all the contents of computer's physical memory to a file, memdump.raw in this case. Memory dump from infected workstation is available in folder `/home/enisa/Desktop/Training-Material/Dig_For_and_Handl_Ev`.

[2]mdd tool can be found at `http://sourceforge.net/projects/mdd/`

3 Evaluate gathered evidence

After we have completed the network traffic analysis, we would like to verify the existence of malicious code from the information gathered. Although we expect to find ZeuS code based on traffic pattern experienced, we must obtain a proof, a code sample in this case. Some malware might be injected to a running system from the network without writing its code to the disk. This is not the way Zeus works, but we've not yet proved it was Zeus, this is our guess.

In the first step we should find the process which was communicating with CC server. We will use VF (Volatility Framework) commands: connections, connscan, sockets and sockscan. Note that we have an updated version of volatility installed. Thus, instead of "**vol.py**´´, simply use "**volatility**´´ command.

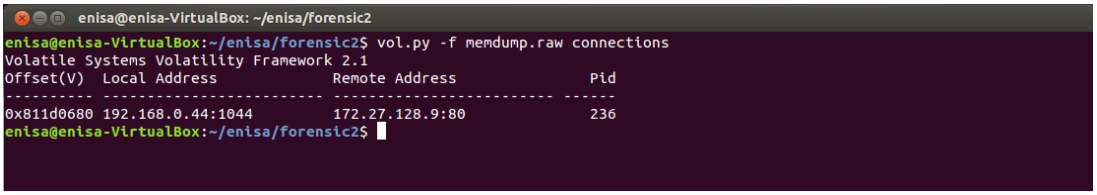

Figure 9: Identification of active connections.

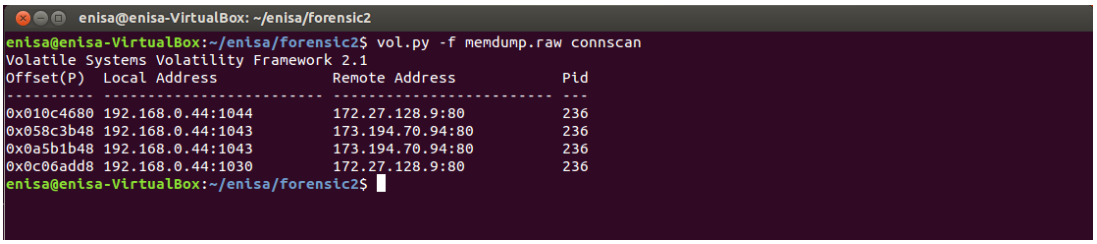

Figure 10: Identification of all connections.

We can see that the computer had an active connection to the IP address 172.27.128.9 (suspected to be a CC server) on port 80 in the very moment of performing memory dump. Additionally, a connection scan found information about past connections, but still not overwritten by other data. Again there were connections to the suspect, along with the IP 173.194.70.94 port 80 – which is Google.com server. You can check back with the Wireshark output in Task 2 to compare.

Another piece of information we extracted is the windows process ID (PID) that opened the connections: 236. We use the Volatility Framework[3] command pslist:

[3]See: `https://www.volatilesystems.com/default/volatility` and `https://code.google.com/p/volatility/wiki/CommandReference22`

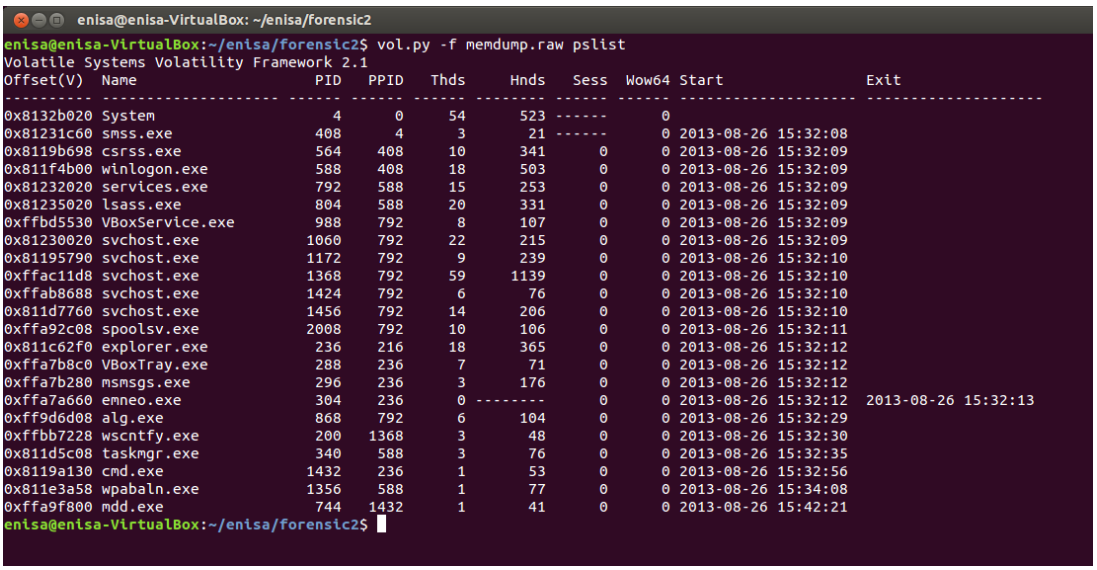

Figure 11: Listing processes.

In the PID column we quickly find that process number 236 is explorer.exe. This is the name of a Windows Explorer executable, the main process of the Windows user interface. Under normal conditions the process should not make any external connections, so we might expect that the binary was modified or some other process injected malicious code into one of explorer's threads. Such an injection is a popular technique used by malicious software to hide its existence.

Another finding is easy to spot here, a process 'emneo.exe' which started and quit almost immediately (this is the only finished process, which makes it easy to find).

Next, we want to find so called 'API hooks'. Hooking is method for modifying a program at runtime. An API hook is basically a way of asking the operating system to call a specific function every time something happens in the system (for example a specific API function is called). One good example is an antivirus registering to be called every time the system or any application wants to open a file from disk. Given the power hooking gives to applications and its creators it's understandable that malicious software uses this technique often.

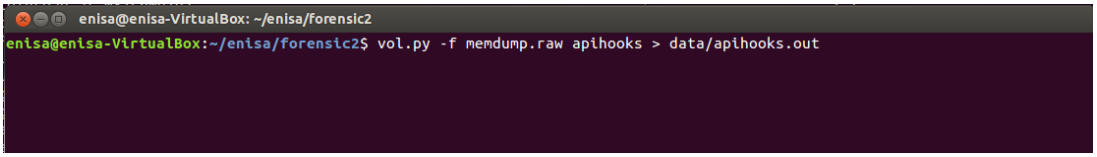

Figure 12: Finding hooks.

This command can take several minutes to execute and can produce large amount of output, so we redirected the output to a file 'apihooks.out'.

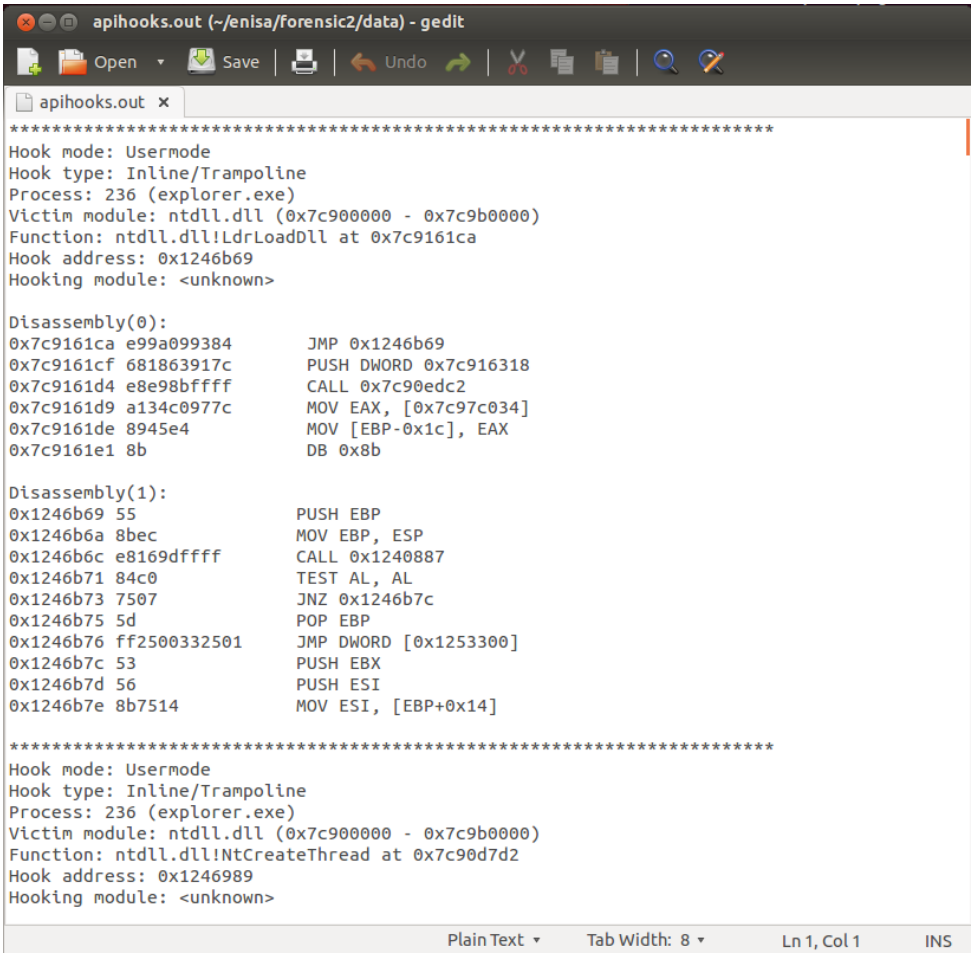

Figure 13: Investigating hooks.

As we can see after opening the file there are many hooks created at miscellaneous API functions calls. Let's find what processes memory spaces contained API hooks with something registered:

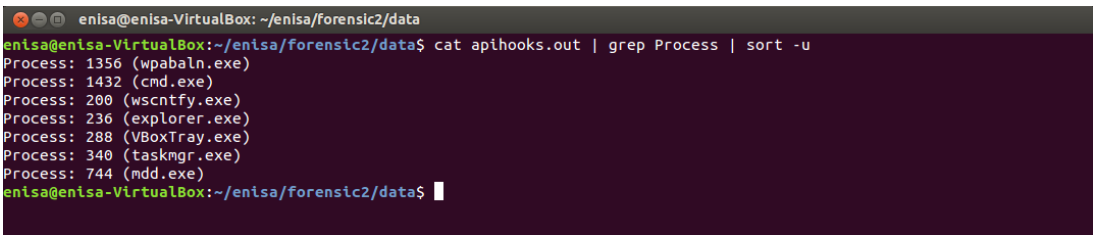

Figure 14: Listing hooks with registered processes.

As we expected explorer.exe is in that list. We will now extract from the computer memory image the memory space of explorer.exe:

Figure 15: Creating process explorer.exe memory dump.

With the extracted memory space and information gathered previously we search the process memory

for phrases of interest. First, is there any reference to the bank we are working for, to the CC server or domain and if we're able to find the mysterious 'emneo.exe'? We will use an internal tool in UNIX system – 'strings', to extract information from a binary file:

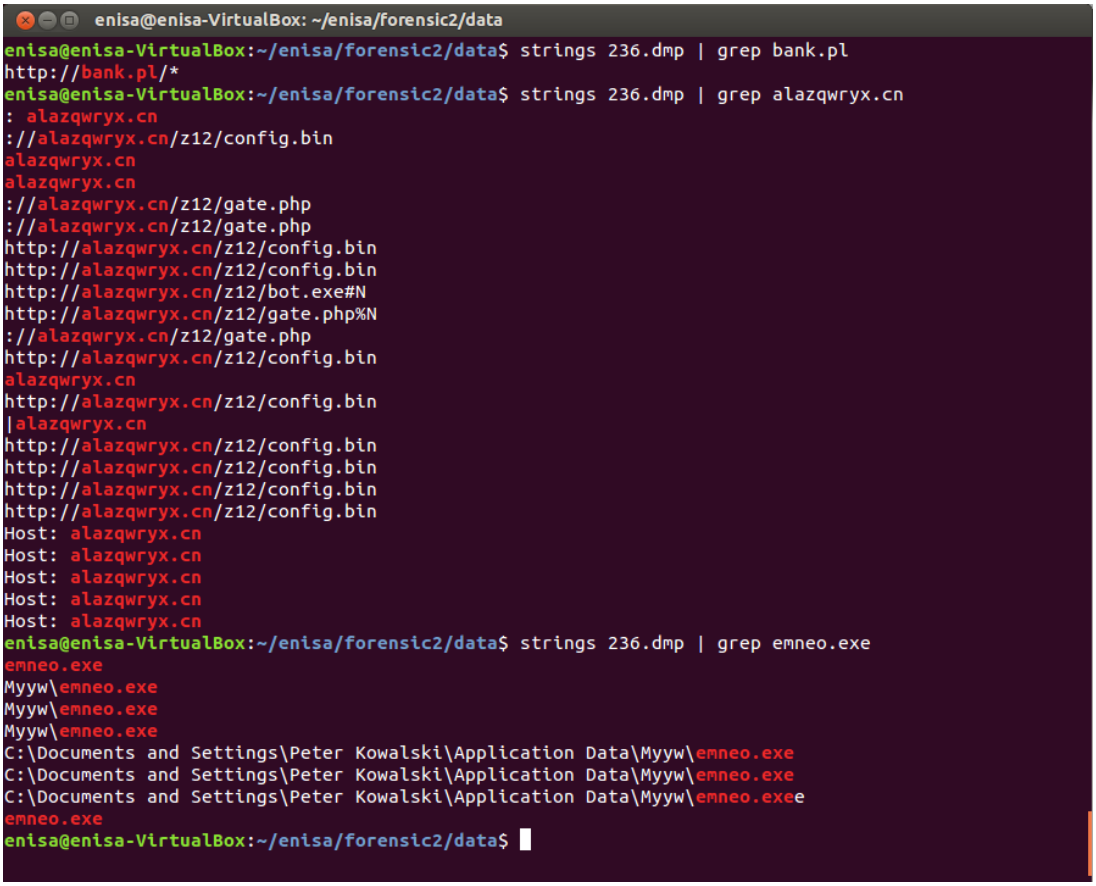

Figure 16: Investigating explorer.exe memory dump.

In all cases the strings we were looking for are present in the memory dump. There's our bank website, there are numerous references to alazqwryx.cn domain and URLs within it. Finally there is a reference to 'emneo.exe' file. We suspect that this is a process run just after powering up the system and then used to inject malicious code into explorer.exe process.

We can now look into the file with the GHex hex editor (accessible through the start button menu) and search for our bank website (we know there is exactly one reference):

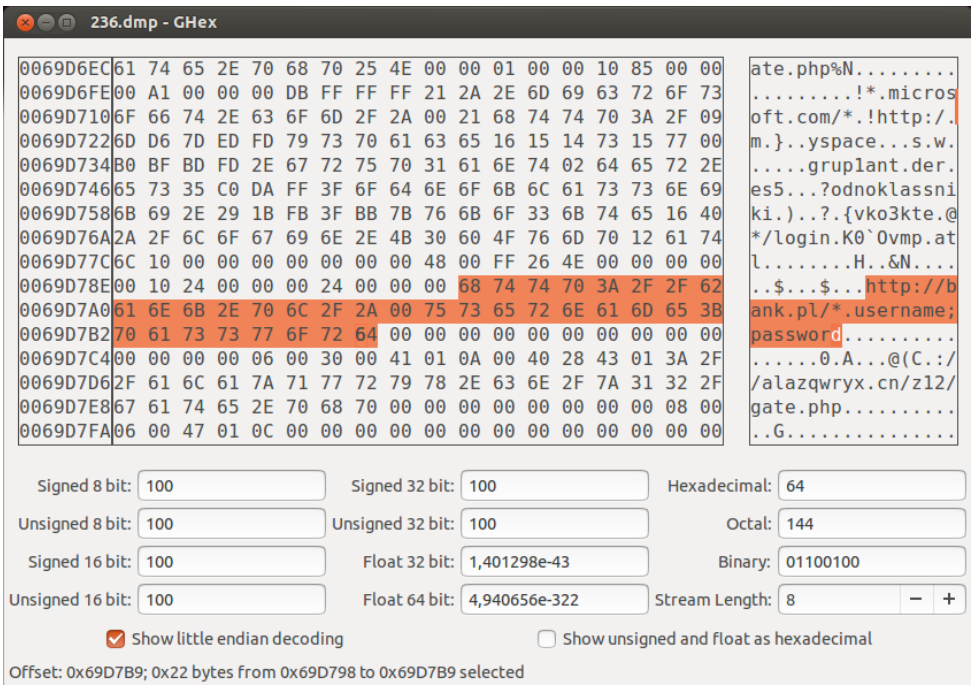

Figure 17: Using hex-editor.

We found a suspicious line, which we expect to be e-banking URL along with part of the input form instructing the malicious code where to inject its scripts. Now we may consider the evidence verification complete, we found traces of malicious software, we found proof of connections being made. However, taking a full image of user computer's hard disk might not be possible or practical. We may want to do one more thing in that case — find and copy files related to the malware found. There is a prepared list of files on the examined system in `/home/enisa/Desktop/Training-Material/Dig_For_and_Handl_Ev/fs.list`. We verify the location of emneo.exe file:

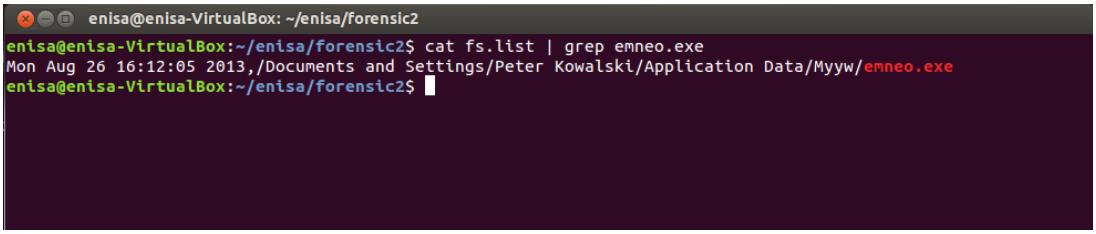

Figure 18: Finding suspected malware locations.

This is exactly the location we have found in the explorer.exe memory dump. We notice the date and time of file creation: 26th August 16:12:05 2013. This is probably the moment this computer got infected, or the moment malware was last updated. We look for other files created around the same time:

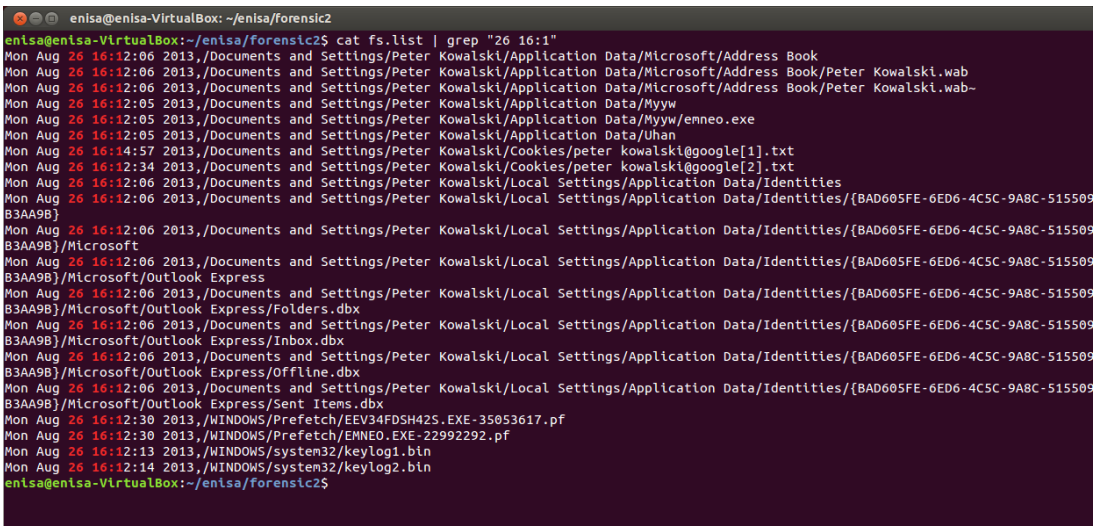

Figure 19: Listing all files suspected to be malware related.

There are more suspicious files found, there are two files in \WINDOWS\system32 – keylog1.bin and keylog2.bin. Their names suggest they may contain information about keys pressed by the user. All the files found should be secured for further investigation.