INSTITUTO SUPERIOR TÉCNICO

Departamento de Engenharia Informática

Forensics Cyber Security

MEIC / METI 2018-2019 – 1st Semester

Digital Forensics Report

Ana Beatriz Nogueira	João Silveira	Martim Zanatti
82433	80789	82517

Overview

For our investigation, we used a persistent install of Kali Linux so we could explore the files on a forensically sound machine. Using md5sum, we verified the MD5 fingerprint of both artifacts in order to ensure their integrity.

```
root@kali:~/lab2# md5sum sally_mem sally_disk
8864691bed9d3712894ea0eff8f21f2e  sally_mem
382c7ae1e99380601ec3bffbe762f60d  sally_disk
```

We include all artifacts collected in **Section 5** of this report with their respective MD5 fingerprints.

Our first step was analysing the disk, listing the partition table with command mmls. From here we could understand that the Linux partition started at position **2048** so we focused on investigating this part of the disk.

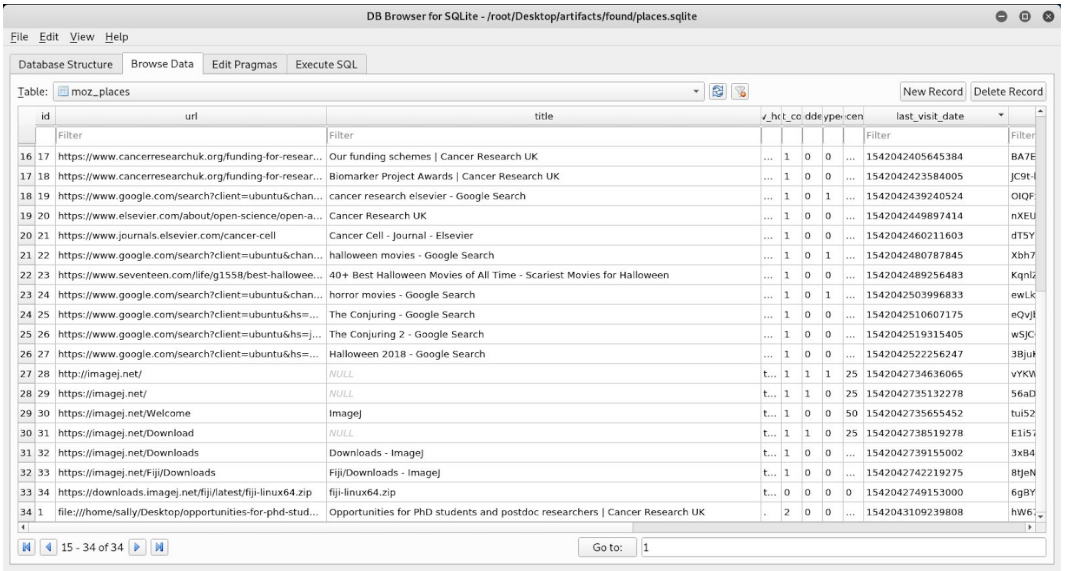
```
root@kali:~/lab2# mmls sally_disk
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0019922943   0019920896   Linux (0x83)
003:  -------   0019922944   0019924991   0000002048   Unallocated
004:  Meta      0019924990   0020969471   0001044482   DOS Extended (0x05)
005:  Meta      0019924990   0019924990   0000000001   Extended Table (#1)
006:  001:000   0019924992   0020969471   0001044480   Linux Swap / Solaris x86 (0x82)
007:  -------   0020969472   0020971519   0000002048   Unallocated
```

We then gave the most importance to the files and folders contained in Sally's home directory.

```
root@kali:~/lab2# fls -o 2048 sally_disk 131076
d/d 569698:     sally
root@kali:~/lab2# fls -o 2048 sally_disk 569698
r/r 569699:     .bashrc
r/r 569700:     .bash_logout
r/r 569701:     examples.desktop
r/r 569702:     .profile
d/d 395150:     .cache
```

We began our investigation with an initial analysis of Sally's disk, exploring various possibilities as to how the malware might have infiltrated her computer. We drew our initial theories based on Sally's report, focusing on the tasks she had performed that day. We decided to verify her claims and see if we could uncover anything.

Sally said she had browsed the web so we analysed her browser history to see if something suspicious came up. Determining her browser wasn't too difficult, simply analysing her disk with **fls** would reveal the **.mozilla** directory on her home folder, which had all of her Firefox logs *(artifact: sally_firefox.zip)*. The browser history itself was stored inside a database file called **places.sqlite**, in the **moz_places** table.



As the logs show, she browsed some pages about halloween movies, cancer research and then downloaded the ImageJ utility. We considered ImageJ to be a possible source of wrongdoing, as the program could have been downloaded from an unreliable source, or to have some malware embedded in the source code. This turned out not to be the case - we checked the victim's downloaded files and we concluded that ImageJ was not the source of the malware - a quick research confirmed the legitimacy of this software and it appeared to have been downloaded from reliable sources.

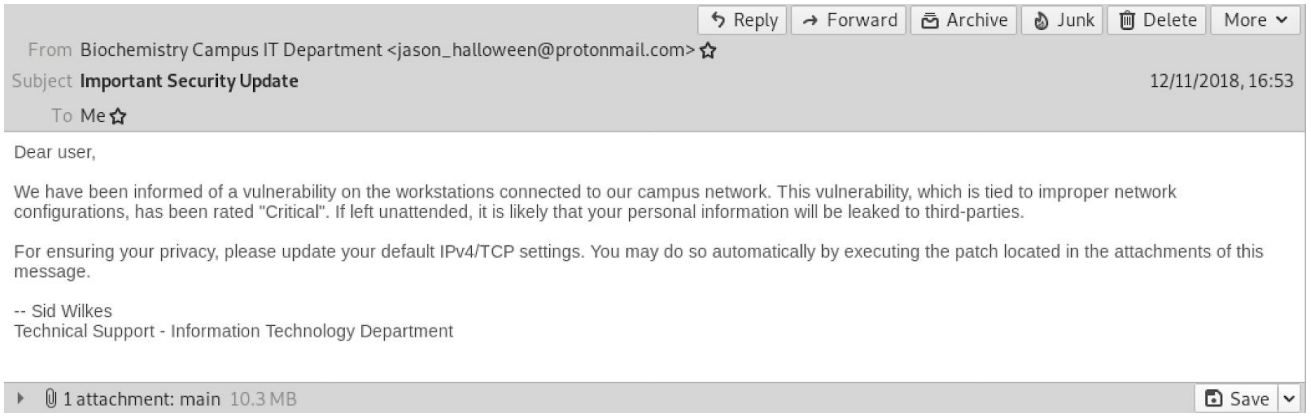
Following Sally's account of events, we decided to check her email. There was no record of her having visited a web email client from her browser. But something interesting found on her disk was a Thunderbird folder **(/home/sally/.thunderbird)**, containing a client profile. Thunderbird is an email application, and by having access to a profile we also had access to the victim's email account *(artifact: sally_thunderbird.zip)*. So we extracted the whole profile from disk by running the following:

```
root@kali:~/Desktop/artifacts# tsk_recover -a -f ext4 -o 2048 -d 575984 sally_disk thunderbird_sally/
Files Recovered: 62
```

After exploring the recovered files, we discovered the **INBOX** file in subfolder **ImapMail/imap.gmail-1.com.**

We decided to install Sally's profile locally on Thunderbird, so we could see clearly what emails she had received. Turns out most of inbound emails were automated, from website subscriptions, newsletters and the like, with the exception of emails from Diogo Barradas (dmbb84@gmail.com), to whom Sally replied.

Going through her inbox, we also uncovered a most suspicious email from jason_halloween@protonmail.com.



It was a clear red flag that this email had an attachment. Another red flag was the disparity in the sender and the actual email address. Sender name claimed to be from the Biochemistry Campus IT Department, and the email was signed by Sid Wilkes, but the email address appeared to have no relation to any of those entities whatsoever. The relation between the popup window from the ransomware and this email address was also clear.

We then extracted the attachment by the name of **main** *(artifact: main)* and **file** revealed that it was an **ELF 64-bit LSB executable**. The strings command itself did not tell much about the inner workings of the program, except for its usage of some cryptographic modules, which strengthened our theory that this executable was the source of the attack.

Our focus shifted to trying to recover the key used to encrypt the files, so we continued analysing the disk in the hopes of finding it. But we found nothing so we explored the memory dump instead.

By running volatility's **linux_proc_maps** on Sally's memory, we had extra information on the **main** process - stack, heap, libraries. Next step was to extract the memory of the presented process mappings with volatility command **linux_dump_map**. The output for the main process alone was extensive, so we focused our attention on the mappings that appeared most suspicious, namely the ones with inode marked as 0 (as the other ones were mostly related to the libraries used by the program) and that had read and write permission *(artifact: main_process_maps.txt)*.

```
root@kali:~/lab2# python ../volatility/vol.py -f sally_mem --profile=LinuxUbuntu160405x64 linux_proc_maps -p 14921
Volatility Foundation Volatility Framework 2.6
Offset             Pid      Name                 Start              End                Flags              Pgoff Major  Minor  Inode      File Path
------------------ -------- -------------------- ------------------ ------------------ ------ ------------------ ------ ------ ---------- ---------
0xffff91d5b56b0000    14921 main                 0x0000000000400000 0x0000000000408000 r-x                0x0      8      1     529882 /home/sally/Downloads/main
0xffff91d5b56b0000    14921 main                 0x0000000000607000 0x0000000000608000 rw-             0x7000      8      1     529882 /home/sally/Downloads/main
0xffff91d5b56b0000    14921 main                 0x0000000000608000 0x0000000000618000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x0000000000ce1000 0x000000000137b000 rw-                0x0      0      0          0 [heap]
0xffff91d5b56b0000    14921 main                 0x00007f127e66d000 0x00007f127e726000 r--                0x0      8      1     409560 /usr/share/fonts/truetype/dejavu/DejaVuSans.ttf
0xffff91d5b56b0000    14921 main                 0x00007f127e726000 0x00007f127e727000 ---                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f127e727000 0x00007f127ef27000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f127ef27000 0x00007f127ef2c000 r-x                0x0      8      1     145504 /usr/lib/x86_64-linux-gnu/libXfixes.so.3.1.0
0xffff91d5b56b0000    14921 main                 0x00007f127ef2c000 0x00007f127f12b000 ---             0x5000      8      1     145504 /usr/lib/x86_64-linux-gnu/libXfixes.so.3.1.0
0xffff91d5b56b0000    14921 main                 0x00007f127f12b000 0x00007f127f12c000 r--             0x4000      8      1     145504 /usr/lib/x86_64-linux-gnu/libXfixes.so.3.1.0
0xffff91d5b56b0000    14921 main                 0x00007f127f12c000 0x00007f127f12d000 rw-             0x5000      8      1     145504 /usr/lib/x86_64-linux-gnu/libXfixes.so.3.1.0
0xffff91d5b56b0000    14921 main                 0x00007f127f12d000 0x00007f127f136000 r-x                0x0      8      1     140838 /usr/lib/x86_64-linux-gnu/libXcursor.so.1.0.2
0xffff91d5b56b0000    14921 main                 0x00007f127f136000 0x00007f127f335000 ---             0x9000      8      1     140838 /usr/lib/x86_64-linux-gnu/libXcursor.so.1.0.2
0xffff91d5b56b0000    14921 main                 0x00007f127f335000 0x00007f127f336000 r--             0x8000      8      1     140838 /usr/lib/x86_64-linux-gnu/libXcursor.so.1.0.2
0xffff91d5b56b0000    14921 main                 0x00007f127f336000 0x00007f127f337000 rw-             0x9000      8      1     140838 /usr/lib/x86_64-linux-gnu/libXcursor.so.1.0.2
0xffff91d5b56b0000    14921 main                 0x00007f127f34f000 0x00007f127f3a3000 r--                0x0      8      1     409562 /usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf
0xffff91d5b56b0000    14921 main                 0x00007f127f3a3000 0x00007f127f3ae000 r--                0x0      8      1     433546 /var/cache/fontconfig/945677eb7aeaf62f1d50efc3fb3ec7d8-le64.cache-6
0xffff91d5b56b0000    14921 main                 0x00007f127f3ae000 0x00007f127f3b4000 r--                0x0      8      1     433528 /var/cache/fontconfig/2cd17615ca594fa2959ae173292e504c-le64.cache-6
0xffff91d5b56b0000    14921 main                 0x00007f127f3b4000 0x00007f127f3bf000 r-x                0x0      8      1     268987 /lib/x86_64-linux-gnu/libnss_files-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f3bf000 0x00007f127f5be000 ---             0xb000      8      1     268987 /lib/x86_64-linux-gnu/libnss_files-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f5be000 0x00007f127f5bf000 r--             0xa000      8      1     268987 /lib/x86_64-linux-gnu/libnss_files-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f5bf000 0x00007f127f5c0000 rw-             0xb000      8      1     268987 /lib/x86_64-linux-gnu/libnss_files-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f5c0000 0x00007f127f5c6000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f127f5c6000 0x00007f127f5d1000 r-x                0x0      8      1     268997 /lib/x86_64-linux-gnu/libnss_nis-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f5d1000 0x00007f127f7d0000 ---             0xb000      8      1     268997 /lib/x86_64-linux-gnu/libnss_nis-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f7d0000 0x00007f127f7d1000 r--             0xa000      8      1     268997 /lib/x86_64-linux-gnu/libnss_nis-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f7d1000 0x00007f127f7d2000 rw-             0xb000      8      1     268997 /lib/x86_64-linux-gnu/libnss_nis-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f7d2000 0x00007f127f7e8000 r-x                0x0      8      1     268981 /lib/x86_64-linux-gnu/libnsl-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f7e8000 0x00007f127f9e7000 ---            0x16000      8      1     268981 /lib/x86_64-linux-gnu/libnsl-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f9e7000 0x00007f127f9e8000 r--            0x15000      8      1     268981 /lib/x86_64-linux-gnu/libnsl-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f9e8000 0x00007f127f9e9000 rw-            0x16000      8      1     268981 /lib/x86_64-linux-gnu/libnsl-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f9e9000 0x00007f127f9eb000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f127f9eb000 0x00007f127f9f3000 r-x                0x0      8      1     268983 /lib/x86_64-linux-gnu/libnss_compat-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127f9f3000 0x00007f127fbf2000 ---             0x8000      8      1     268983 /lib/x86_64-linux-gnu/libnss_compat-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127fbf2000 0x00007f127fbf3000 r--             0x7000      8      1     268983 /lib/x86_64-linux-gnu/libnss_compat-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127fbf3000 0x00007f127fbf4000 rw-             0x8000      8      1     268983 /lib/x86_64-linux-gnu/libnss_compat-2.23.so
0xffff91d5b56b0000    14921 main                 0x00007f127fbf4000 0x00007f127fc74000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f127fc74000 0x00007f127fc75000 r-x                0x0      8      1      22542 /tmp/_MEIlXS6RU/future_builtins.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f127fc75000 0x00007f127fe74000 ---             0x1000      8      1      22542 /tmp/_MEIlXS6RU/future_builtins.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f127fe74000 0x00007f127fe75000 r--                0x0      8      1      22542 /tmp/_MEIlXS6RU/future_builtins.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f127fe75000 0x00007f127fe76000 rw-             0x1000      8      1      22542 /tmp/_MEIlXS6RU/future_builtins.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f127fe76000 0x00007f127fed4000 r-x                0x0      8      1      22561 /tmp/_MEIlXS6RU/libssl.so.1.0.0
0xffff91d5b56b0000    14921 main                 0x00007f127fed4000 0x00007f12800d4000 ---             0x5e000      8      1      22561 /tmp/_MEIlXS6RU/libssl.so.1.0.0
0xffff91d5b56b0000    14921 main                 0x00007f12800d4000 0x00007f12800d8000 r--             0x5e000      8      1      22561 /tmp/_MEIlXS6RU/libssl.so.1.0.0
0xffff91d5b56b0000    14921 main                 0x00007f12800d8000 0x00007f12800df000 rw-             0x62000      8      1      22561 /tmp/_MEIlXS6RU/libssl.so.1.0.0
0xffff91d5b56b0000    14921 main                 0x00007f12800df000 0x00007f12800f4000 r-x                0x0      8      1      22537 /tmp/_MEIlXS6RU/_ssl.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f12800f4000 0x00007f12802f3000 ---             0x15000      8      1      22537 /tmp/_MEIlXS6RU/_ssl.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f12802f3000 0x00007f12802f4000 r--             0x14000      8      1      22537 /tmp/_MEIlXS6RU/_ssl.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f12802f4000 0x00007f12802f8000 rw-             0x15000      8      1      22537 /tmp/_MEIlXS6RU/_ssl.x86_64-linux-gnu.so
0xffff91d5b56b0000    14921 main                 0x00007f12802f8000 0x00007f12803b8000 rw-                0x0      0      0          0
0xffff91d5b56b0000    14921 main                 0x00007f12803b8000 0x00007f12803ba000 r-x                0x0      8      1      22560 /tmp/_MEIlXS6RU/resource.x86_64-linux-gnu.so
```

The process mapping with start address **0x7f127fbf4000** was where we found the key, once again with the help of **strings**. *(artifacts: task.14921.0x7f127fbf4000.vma , 7f127fbf4000_strings.txt)*

```
root@kali:~/lab2# python ../volatility/vol.py -f sally_mem --profile=LinuxUbuntu160405x64 linux_dump_map -p 14921 -s 0x00007f127fbf4000 --dump-dir=14921_rw/
Volatility Foundation Volatility Framework 2.6
Task       VM Start          VM End            Length Path
---------- ----------------- ----------------- ------ ----
     14921 0x00007f127fbf4000 0x00007f127fc74000            0x80000 14921_rw/task.14921.0x7f127fbf4000.vma
```

The key found was **47683b9a9663c065353437b35c5d8519**. In **Section 2** of the report we explain in detail how we verified the correctness of the key and how we managed to decrypt the files.

Three peculiar things about the excerpt of the strings output (presented in the image to the right):

```
f02d0
echo "47683b9a9663c065353437b35c5d8519" > key.txt
[PEXPECT]$
 unset PROMPT_COMMAND
jason@optiplex:~$ PS1='[PEXPECT]\$ '
fcntl() argument 3 must be string or read-only buffer, not int
echo "47683b9a9663c065353437b35c5d8519" > key.txt
[PEXPECT]$
/tmp/_MEIlXS6RU/encodings/utf_8.pyc
/home/sally/Downloads/main.py
```

- the key is redirected to a **key.txt** file
- the **jason@optiplex:~** prompt - hinting at an ssh connection to a remote machine
- the apparent usage of **Pexpect** - from the docs: *"Pexpect is a pure Python module for spawning child applications; controlling them; (...) Pexpect can be used for automating interactive applications such as ssh, ftp, passwd, telnet, etc."*

From this, we theorised that the attacker wanted to send the key back to himself. As a result, we took a look at some of the network oriented commands of the volatility toolkit, starting with **linux_netscan** *(artifact: mem_netscan.txt)*:

```
91d5b56fb000 TCP       10.0.2.15       :50924 216.58.214.174  :   80 CLOSE
91d5b56fb800 TCP       10.0.2.15       :34382 144.92.48.177   :  443 CLOSE
91d5b5e50000 TCP       10.0.2.15       :35512 146.193.41.57   :   22 CLOSE
91d5b5e50800 TCP       10.0.2.15       :54410 54.187.46.234   :  443 CLOSE
```

We analysed the 6th column first, corresponding to the destination address port - this would give us pointers as to what sort of traffic was being sent out of the computer. Some ports were more suspicious than others in the context of the investigation:

- Ports **80** (HTTP), **443** (HTTPS) and **993** (IMAP over SSL/TLS) are relatively harmless, considering Sally's account of her activities that day - browsing the web (both with HTTP and HTTPS as we've established with her browser history) and checking her email (according to netscan, IMAP packets were sent to Google servers, which makes sense since Sally's email provider is gmail).
- **22 (SSH)**, **631 (IPP - Internet Printing Protocol)** could be considered suspicious, since the attacker could exfiltrate information through them.

We investigated the communication with port 631, linked to the **cups** process also running on the system but it led us nowhere.

The **linux_lsof** command also indicates some socket activity related to the **main** process. *(artifact: mem_lsof.txt)*

1 Can you determine how the malware has taken over Sally's computer?

Yes. As mentioned in the section above, we discovered that the source of the malware was the executable attachment found on the email Sally received from jason_halloween@protonmail.com.

```
root@kali:~/Desktop/artifacts# fls -o 2048 sally_disk 573442
d/d 576046:     LiME
r/r * 529882(realloc):  main
d/d 529881:     Fiji.app
root@kali:~/Desktop/artifacts# istat -o 2048 sally_disk 529882
inode: 529882
Allocated
Group: 64
Generation Id: 689348453
uid / gid: 1000 / 1000
mode: rrw-rw-r--
Flags: Extents,
size: 94
num of links: 1

Inode Times:
Accessed:       2018-11-12 17:24:38.038254458 (WET)
File Modified:  2018-11-12 17:24:38.094254576 (WET)
Inode Modified: 2018-11-12 17:24:38.094254576 (WET)
File Created:   2018-11-12 17:24:38.038254458 (WET)

Direct Blocks:
2191067
```

We confirmed this by checking if the file had been downloaded and accessed by Sally. With **fls** we found that a file named **main** had previously existed in the Downloads folder.

In addition to that we had all the memory logs from the processes running in the victim's system. Running volatility command **linux_pslist** shows the time at which the process started executing, and also that if forked a new process.

```
0xffff91d5b56b5b00 main                14919          1211           1000           1000   0x00000000317ce000 2018-11-12 17:15:45 UTC+0000
0xffff91d5b56b0000 main                14921          14919          1000           1000   0x000000002fb0a000 2018-11-12 17:15:45 UTC+0000
```

The analysis reported in the previous section applies here - all the crypto modules imported, volatility commands outputs, etc, they all point to the **main** process being the culprit.

Shortly after that the files were encrypted.

2 Can you recover Sally's original files? If you do not succeed at retrieving the original files, can you at least extract some of its fragments?

Yes. We were able to extract a key that supposedly had been used to encrypt the files (extraction described in the Overview section of the report).

We then moved on to confirm if this was indeed the correct key. From executing **strings main** we were aware of some of the libraries used to encrypt the files, so we wrote a program using those same libraries that would decrypt the files *(artifact: aes.py)*.

```
root@kali:~/Desktop/artifacts/found# python3 aes.py 47683b9a9663c065353437b35c5d8519 Documents/
cancer_cells/                          Image_Processing_with_ImageJ.pdf.encrypted  paper_draft.txt.encrypted
Image_Processing_with_ImageJ.pdf       paper_draft.txt
root@kali:~/Desktop/artifacts/found# python3 aes.py 47683b9a9663c065353437b35c5d8519 Documents/paper_draft.txt.encrypted
```

To run the script execute ***python3 aes.py key filename***. The script expects the file to have suffix *.encrypted* and writes the output of the decryption to a new file with the same name as *filename* but without the *.encrypted* suffix. We ran the script over all encrypted files with the key we had previously found and were able to recover all of them *(artifact: sally_files.zip)*.

All the recovered files seem to make sense and be Sally's original files. However we would need the victim's confirmation in order to know for certain that these files are the ones Sally lost access to.

3 What can you tell about the identity of the attacker?

We know that the attacker is the owner of the jason_halloween@protonmail.com address.

We checked the address of the machine that had connected to Sally's through SSH with **whois 146.193.42.57**.

```
% Information related to '146.193.32.0/19AS5516'

route:          146.193.32.0/19
descr:          INESC - Instituto de Engenharia de Sistemas e Computadores
descr:          Lisboa, Portugal
origin:         AS5516
mnt-by:         INESC-MNT
created:        2004-04-05T12:34:27Z
last-modified:  2007-03-16T17:33:57Z
source:         RIPE # Filtered
```

4 Elaborate a timeline of the most significant events of the case.

12th of November 2018

- **4:53 PM**: Email with the ransomware arrives (sent by jason_halloween@protonmail.com)
- **5:05 PM**: Earliest record of Sally browsing the web (Google search for cancer research)
- **5:11 PM**: Sally replies to Diogo Barradas email about movie night
- **5:12 PM**: Sally downloads ImageJ
- **5:15 PM**: *main* processes start executing
- **5:18 PM**: html file related to PhD opportunities is accessed (file last accessed and Firefox history)
- **5:20 PM**: Files are already encrypted at this time (last modified on *istat*)
- **5:20 PM**: sally_mem file was created
- **5:24 PM**: Last modification of .bash_history - possibly when memory dump was made and sent to machine *dmbb@turbina.gsd.inesc-id.pt*

5 Artifacts

We present below the list of the most relevant files found and produced during the investigation, a short description of their contents and their respective MD5 fingerprints.

File name	File contents	MD5
sally_firefox.zip	Zip containing Sally's Firefox data, inside which is found the **places.sqlite** file with her browser history.	a42b4c8137b7addc82998b249da6e260
sally_thunderbird.zip	Zip with Sally's Thunderbird email profile. Contains all her email communication through address **jones.sally1993@gmail.com**.	04e5879c8e138fcef7bcd1f6fcefa4f4
main	Executable attachment sent to Sally via email and source of the malware.	324ddc336159dd62e182e3abf12c9b0a
main_proc_maps.txt	All of the possible process mappings for the main process (pid 14919 and 14921). Obtained with linux_proc_maps.	a24249554b260415f9baed9c5ee20f09
task.14921.0x7f127fbf4000.vma	Memory segment of process 14921, where the decryption key was found.	ef0792406d89d291c0285c679a11cb21
7f127fbf4000_strings.txt	Output of **strings** over the above file. Contains the decryption key.	72d180371a9351b38ab9843700bac771
mem_netscan.txt	Output of **linux_netscan** from examining the sally_mem artifact.	a1ce5868838b140e6eb1dd433e1cc2a3
mem_lsof.txt	Output of **linux_lsof** from examining the sally_mem artifact.	1ac928f7b87f3bf89026ac889096db56
aes.py	Script used to decrypt the files on Sally's Documents folder.	55d73a8ab052be1386a744e76b732211

sally_files.zip	Zip file containing Sally's Documents folder with the files we recovered after decrypting.	`c6d01a7f85e8f9760fce4ec93dfb88c4`