ZEIT8028: Digital Forensics

Lab 2: Disk Forensics

Background

You're still feeling quite anxious being the only junior analyst employed by the small digital forensic investigation and expert witness consultancy. Although you've been working hard researching Microsoft Windows forensic artefacts, you're itching for your first investigation and to prove your worth.

"Hey you, the new one", your boss yells from across the floor, *"Get in here."* You nervously get up from your desk and walk into your boss's office, dreading what she might have in store for you. *"How would you like a case?"*, your boss coyly asks. You try to play it cool, but you can't contain your excitement, cracking a slight smile, *"Sure, what have you got?"*

Your company has been contracted to perform an investigation into the breach of corporate-in-confidence material at **M57.biz**. The customer informs you that a spreadsheet containing sensitive personnel information has been posted to a competitor's website. The customer believes that Jean, the Chief Financial Officer (CFO), is responsible for the leak. During an internal interview with both Jean and Alison (the President) the following statements were made:

Jean (CFO)

- *"Alison asked me to prepare the spreadsheet as part of a new funding round."*
- *"Alison asked me to send the spreadsheet to her by email."*
- *"That's all I know..."*

Alison (President)

- *"I don't know what Jean is talking about."*

- *"I never asked Jean for the spreadsheet."*
- *"I never received the spreadsheet by email."*

As the matter couldn't be resolved internally, the customer contracted your company to conduct a thorough digital forensic investigation.

Specifically, the customer wants you to answer the following questions in a written report:

- When did Jean create the spreadsheet?
- How did it get from her computer to the competitor's website?
- Who else from the company was involved?

Evidence

For this lab you've been provided one (1) raw disk image which is all the evidence you require to complete your investigation.

Before you start the first exercise, verify that your evidence isn't corrupt. This ensures that you don't waste your time and effort troubleshooting data that isn't working as expected.

The pertinent evidence is available in the `Lab 2 - Disk Forensics.7z` course bundle and it's metadata is as follows:

```
FILE:   disk.bin
SIZE:   10,737,418,240 bytes
SHA1:   ba7dc57e08bb6e3393aee15c713ae04feadcd181
MD5:    78a52b5bac78f4e711607707ac0e3f93
```

To unzip the 7z archive, you should be able to use the built-in archive utility on your OS. Alternatively, here are some other options:

- On macOS, use Keka
- On the course OVA or other Linux distro, use 7z: `7z x "Lab 2 - Disk Forensics.7z"`
- On windows, use 7zip

Things to Remember

During your investigation, remember to take lots of notes and document everything that you find. Doing so will make your life significantly easier as you start to bring together your smaller analytical discoveries into a larger picture and will prevent you from questioning or repeating analysis. Most importantly, doing so will make your task of compiling a final report much easier.

Additionally, there are several ways to get the evidence into your VM. Follow the steps in Lab 1: Virtual Analysis Environment to set up a shared folder and/or a secondary hard disk. You'll get the

best performance by doing both, then copying the data from your shared folder to your additional disk, then working on it from there. Alternatively, put the evidence on a USB drive and attach that to your VM (an external SSD would be best).

Exercise 1: Preparation

Your first exercise is to prepare your evidence for analysis.

You must determine what kind of data has been given to you by the acquisition team. You suspect it's a disk image, but what kind of disk image is it?

a) Using the sleuthkit (tsk) utility, `mmls`, determine and analyse the layout of the disk.

> ▾ **Hint**
>
> If you're not familiar with `mmls`, refer to its usage documentation:
>
> ```
> analyst@forensics~$ man 1 mmls
> ```

> ▾ **Solution**
>
> This part of the lab is fairly straightforward, all you really need to do is run `mmls` on the binary image:
>
> ```
> analyst@forensics~$ mmls disk.bin
> 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   0000000062   0000000063   Unallocated
> 002:  000:000   0000000063   0020948759   0020948697   NTFS / exFAT (0x07)
> 003:  -------   0020948760   0020971519   0000022760   Unallocated
> ```
>
> However, this won't always be the case, and you may need some more advanced `mmls` options to find your partition.

b) Locate the filesystem of interest and manually verify the filesystem type using a raw data

parsing utility.

▾ **Hint**

If you're not familiar with `hexdump`, refer to its usage documentation:

```
analyst@forensics~$ man 1 hexdump
```

▾ **Hint**

You can use Python to convert `sectors` to `bytes`:

```
analyst@forensics~$ python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 63*512
32256
>>>
```

▾ **Hint**

Alternatively, if you prefer to avoid using a command-line utility you can use the graphical user interface utility `wxHexEditor` instead, which has already been installed into your analysis environment for you and can be opened from terminal using this command:

```
analyst@forensics~$ wxHexEditor &
```

▾ **Solution**

One way to solve this problem, now that we learned where the primary partition starts (thanks to `mmls`):

```
analyst@forensics~$ hexdump -C -n 512 -s 32256 disk.bin
00007e00  eb 52 90 4e 54 46 53 20  20 20 20 00 02 08 00 00  |.R.NTFS    .....|
00007e10  00 00 00 00 00 f8 00 00  3f 00 ff 00 3f 00 00 00  |........?...?...|
00007e20  00 00 00 00 80 00 80 00  d8 a6 3f 01 00 00 00 00  |..........?.....|
00007e30  00 00 0c 00 00 00 00 00  6d fa 13 00 00 00 00 00  |........m.......|
00007e40  f6 00 00 00 01 00 00 00  1f c2 4f 74 08 50 74 7e  |..........Ot.Pt~|
00007e50  00 00 00 00 fa 33 c0 8e  d0 bc 00 7c fb b8 c0 07  |.....3.....|....|
00007e60  8e d8 e8 16 00 b8 00 0d  8e c0 33 db c6 06 0e 00  |..........3.....|
00007e70  10 e8 53 00 68 00 0d 68  6a 02 cb 8a 16 24 00 b4  |..S.h..hj....$..|
00007e80  08 cd 13 73 05 b9 ff ff  8a f1 66 0f b6 c6 40 66  |...s......f...@f|
00007e90  0f b6 d1 80 e2 3f f7 e2  86 cd c0 ed 06 41 66 0f  |.....?.......Af.|
00007ea0  b7 c9 66 f7 e1 66 a3 20  00 c3 b4 41 bb aa 55 8a  |..f..f. ...A..U.|
00007eb0  16 24 00 cd 13 72 0f 81  fb 55 aa 75 09 f6 c1 01  |.$...r...U.u....|
00007ec0  74 04 fe 06 14 00 c3 66  60 1e 06 66 a1 10 00 66  |t......f`..f...f|
00007ed0  03 06 1c 00 66 3b 06 20  00 0f 82 3a 00 1e 66 6a  |....f;. ...:..fj|
00007ee0  00 66 50 06 53 66 68 10  00 01 00 80 3e 14 00 00  |.fP.Sfh.....>...|
00007ef0  0f 85 0c 00 e8 b3 ff 80  3e 14 00 00 0f 84 61 00  |........>.....a.|
00007f00  b4 42 8a 16 24 00 16 1f  8b f4 cd 13 66 58 5b 07  |.B..$.......fX[.|
00007f10  66 58 66 58 1f eb 2d 66  33 d2 66 0f b7 0e 18 00  |fXfX..-f3.f.....|
```

```
00007f20  66 f7 f1 fe c2 8a ca 66  8b d0 66 c1 ea 10 f7 36  |f......f..f....6|
00007f30  1a 00 86 d6 8a 16 24 00  8a e8 c0 e4 06 0a cc b8  |......$.........|
00007f40  01 02 cd 13 0f 82 19 00  8c c0 05 20 00 8e c0 66  |........... ...f|
00007f50  ff 06 10 00 ff 0e 0e 00  0f 85 6f ff 07 1f 66 61  |..........o...fa|
00007f60  c3 a0 f8 01 e8 09 00 a0  fb 01 e8 03 00 fb eb fe  |................|
00007f70  b4 01 8b f0 ac 3c 00 74  09 b4 0e bb 07 00 cd 10  |.....<.t........|
00007f80  eb f2 c3 0d 0a 41 20 64  69 73 6b 20 72 65 61 64  |.....A disk read|
00007f90  20 65 72 72 6f 72 20 6f  63 63 75 72 72 65 64 00  | error occurred.|
00007fa0  0d 0a 4e 54 4c 44 52 20  69 73 20 6d 69 73 73 69  |..NTLDR is missi|
00007fb0  6e 67 00 0d 0a 4e 54 4c  44 52 20 69 73 20 63 6f  |ng...NTLDR is co|
00007fc0  6d 70 72 65 73 73 65 64  00 0d 0a 50 72 65 73 73  |mpressed...Press|
00007fd0  20 43 74 72 6c 2b 41 6c  74 2b 44 65 6c 20 74 6f  | Ctrl+Alt+Del to|
00007fe0  20 72 65 73 74 61 72 74  0d 0a 00 00 00 00 00 00  | restart........|
00007ff0  00 00 00 00 00 00 00 00  83 a0 b3 c9 00 00 55 aa  |..............U.|
00008000
```

We can see this is an NTFS filesystem.

c) Finally, extract the filesystem from the disk image using the `dd` utility. Document all your findings, including the SHA1 hash of the newly created artefact.

▼ **Hint**

If you're not familiar with `dd`, refer to its usage documentation:

```
analyst@forensics~$ man 1 dd
```

▼ **Solution**

There are several ways to solve this problem. For example:

```
analyst@forensics~$ dd if=disk.bin of=out.bin bs=512 skip=63 count=20948697
status=progress
20948697+0 records in
20948697+0 records out
10725732864 bytes (11 GB, 10 GiB) copied, 4773.66 s, 2.2 MB/s
```

Once this command completes, you can run `file` on the resulting file and confirm the disk contains an NTFS volume:

```
analyst@forensics~$ file out.bin
out.bin: DOS/MBR boot sector, code offset 0x52+2, OEM-ID "NTFS    ",
sectors/cluster 8, Media descriptor 0xf8, sectors/track 63, heads 255, hidden
sectors 63, dos < 4.0 BootSector (0x80), FAT (1Y bit by descriptor); NTFS,
sectors/track 63, sectors 20948696, $MFT start cluster 786432, $MFTMirror start
cluster 1309293, bytes/RecordSegment 2^(-1*246), clusters/index block 1, serial
number 07e745008744fc21f; containsMicrosoft Windows XP/VISTA bootloader NTLDR
```

Because the output is comma separated, we can us `tr` (translate) to convert each comma to a newline character to make the output easier to read:

```
analyst@forensics~$ file out.bin | tr ',' '\n'
out.bin: DOS/MBR boot sector
 code offset 0x52+2
 OEM-ID "NTFS    "
 sectors/cluster 8
 Media descriptor 0xf8
 sectors/track 63
 heads 255
 hidden sectors 63
 dos < 4.0 BootSector (0x80)
 FAT (1Y bit by descriptor); NTFS
 sectors/track 63
 sectors 20948696
 $MFT start cluster 786432
 $MFTMirror start cluster 1309293
 bytes/RecordSegment 2^(-1*246)
 clusters/index block 1
 serial number 07e745008744fc21f; contains bootstrap NTLDR
```

The output file size is `10,725,732,864 bytes`, and the SHA1 is `f8ec0f28b54ef1a9d4f3775c1903bc28493a2743`. As you can see, this command took 4773.66 seconds (~80 minutes) on my VM when I was using shared folders. In contrast, when I used a second disk attached the VM as described in Lab 1, it was orders of magnitude faster:

```
analyst@forensics~$ dd if=disk.bin of=out.bin bs=512 skip=63 count=20948697
status=progress
20948697+0 records in
20948697+0 records out
10725732864 bytes (11 GB, 10 GiB) copied, 46.579 s, 230 MB/s
```

References

https://tinyapps.org/docs/mount_partitions_from_disk_images.html

---

Exercise 2: Initial Inspection

Now that you've obtained a raw copy of the pertinent filesystem from the original image, it's time

to have an initial look at what kind of data it contains.

a) Dynamically inspect the contents of the filesystem by mounting the filesystem you extracted in exercise one to the loop device by using the `mount` utility.

▼ **Hint**

If you're not familiar with `mount`, refer to its usage documentation:

```
analyst@forensics~$ man 8 mount
```

▼ **Solution**

Start by creating a mount location in `/mnt`:

```
analyst@forensics~$ sudo mkdir /mnt/lab2
```

Then, mount the binary disk as read-only (ro), specifying offset 0, ntfs as the partition type, your binary file, and your mount point:

```
analyst@forensics~$ sudo mount -ro loop,offset=0 -t ntfs out.bin /mnt/lab2
```

Then, review the contents of your mounted disk:

```
analyst@forensics~$ ls -lah /mnt/lab2/
total 769M
drwxrwxrwx 1 root root 4.0K Jul 20  2008  .
drwxr-xr-x 4 root root 4.0K Feb 11 09:04  ..
-rwxrwxrwx 1 root root    0 May 13  2008  AUTOEXEC.BAT
-rwxrwxrwx 1 root root  211 May 13  2008  boot.ini
-rwxrwxrwx 1 root root    0 May 13  2008  CONFIG.SYS
drwxrwxrwx 1 root root 4.0K Jul 12  2008 'Documents and Settings'
-rwxrwxrwx 1 root root    0 May 13  2008  IO.SYS
-rwxrwxrwx 1 root root 1.3K Jul 18  2008  IPH.PH
-rwxrwxrwx 1 root root    0 May 13  2008  MSDOS.SYS
-rwxrwxrwx 1 root root  47K Aug  4  2004  NTDETECT.COM
-rwxrwxrwx 1 root root 245K May 14  2008  ntldr
-rwxrwxrwx 1 root root 768M Jul 21  2008  pagefile.sys
drwxrwxrwx 1 root root 8.0K Jul 18  2008 'Program Files'
drwxrwxrwx 1 root root    0 Jul 11  2008  RECYCLER
drwxrwxrwx 1 root root 4.0K May 13  2008 'System Volume Information'
drwxrwxrwx 1 root root  28K Jul 21  2008  WINDOWS
```

You can unmount the filesystem using `umount`:

```
analyst@forensics~$ sudo umount /mnt/lab2
analyst@forensics~$ ls -l /mnt/lab2/
total 0
```

b) Statically inspect the contents of the filesystem using the tsk utility `fls`.

▾ **Hint**

If you're not familiar with `fls`, refer to its usage documentation:

```
analyst@forensics~$ man 1 fls
```

▾ **Solution**

```
analyst@forensics~$ fls out.bin
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-1:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-1:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-144-17:   $Secure:$SDH
r/r 9-144-16:   $Secure:$SII
r/r 9-128-18:   $Secure:$SDS
r/r 10-128-1:   $UpCase
r/r 3-128-3:    $Volume
r/r 7451-128-1: AUTOEXEC.BAT
r/r 3513-128-3: boot.ini
r/r 7450-128-1: CONFIG.SYS
d/d 3519-144-6: Documents and Settings
r/r 7452-128-1: IO.SYS
r/r 27624-128-3:        IPH.PH
r/r 7453-128-1: MSDOS.SYS
r/r 3485-128-3: NTDETECT.COM
r/r 3481-128-3: ntldr
r/r 27-128-1:   pagefile.sys
d/d 3999-144-6: Program Files
d/d 23827-144-1:        RECYCLER
d/d 3522-144-6: System Volume Information
d/d 28-144-6:   WINDOWS
V/V 32848:      $OrphanFiles
```

Take your time to explore the contents of the disk and how the utilities work. What are the benefits and pitfalls of using either of the inspection methods? Is there any risk posed to the evidence when using either method? If so, what are they?

Exercise 3: Commencing Analysis

You should now have a verified dataset and be ready to commence your analysis.

a) Statically locate the sensitive document from Jean's computer using the tsk utility `fls`. Once you've found the file, take note of its inode. Use this inode value to extract the file using the tsk utility `icat`.

▾ **Hint**

If you're not familiar with `icat`, refer to its usage documentation:

```
analyst@forensics~$ man 1 icat
```

▾ **Solution**

One way to achieve this is to search recursively for filenames containing `m57`:

```
analyst@forensics~$ fls -r out.bin | grep -i m57 | less
...
++++++ d/d 30938-144-1: m57jean
++++++ r/r 32714-128-1: m57biz.LNK
+++ r/r 32712-128-3:    m57biz.xls
...
```

We can see the inodes of the files and folders in the middle column. Use `icat` to extract the file:

```
analyst@forensics~$ icat out.bin 32712-128-3 > m57biz.xls
analyst@forensics~$ ls -lah
total 20G
drwxr-xr-x 2 analyst analyst 4.0K Feb 16 01:37 .
drwxr-xr-x 3 analyst analyst 4.0K Feb 16 01:37 ..
-r--r--r-- 1 analyst analyst  10G Jul 10  2019 disk.bin
-rw-r--r-- 1 analyst analyst 285K Feb 16 01:36 m57biz.xls
-rw-r--r-- 1 analyst analyst  10G Feb 16 01:27 out.bin
```

Once extracted, you can inspect the document within your analysis environment using

`gnumeric m57biz.xls` . Although it's interesting to read the content, the real forensic information lies within the document metadata.

Legacy Microsoft Excel spreadsheet files (.xls) are contained within a highly structured file container known as an Object Linking and Embedding (OLE) Compound File (CF). OLECF uses a FAT-like file system to define blocks that are assigned to the stream using multiple allocation tables. It also uses a directory structure to define the name of the streams.

The OLECF is used to store:

1. Microsoft Office 97-2003 documents:
   - Word Document (DOC)
   - Excel Spreadsheet (XLS)
   - Powerpoint Presentation (PPT)

2. MSN (Toolbar) (C:\Documents and Settings\%USERNAME%\Local Settings\Application Data\Microsoft\MSNe\msninfo.dat)
3. Jump Lists
4. StickyNotes.snt
5. Thumbs.db
6. Windows Installer (.msi) and patch file (.msp)
7. Windows Search (srchadm.msc)

b) An OLECF file has the following file signature (as a hexadecimal byte sequence): `d0 cf 11 e0 a1 b1 1a e1` . Verify that the file you have extracted is an OLECF file using a raw data parsing utility (e.g. `hexdump` or `xxd`).

▼ **Solution**

```
analyst@forensics~$ hexdump -C m57biz.xls | head -1
00000000  d0 cf 11 e0 a1 b1 1a e1  00 00 00 00 00 00 00 00  |................|


analyst@forensics~$ xxd m57biz.xls | head -1
00000000: d0cf 11e0 a1b1 1ae1 0000 0000 0000 0000  ................
```

c) As you might expect, OLECF stored Excel files are very metadata rich. Inspect the Excel file metadata using the `file` utility.

▼ **Hint**

If you're not familiar with `file` , refer to its usage documentation:

```
analyst@forensics~$ man 1 file
```

▼ **Solution**

```
analyst@forensics~$ file m57biz.xls | tr ',' '\n'
m57biz.xls: Composite Document File V2 Document
 Little Endian
 Os: Windows
 Version 5.1
 Code page: 1252
 Author: Alison Smith
 Last Saved By: Jean User
 Name of Creating Application: Microsoft Excel
 Create Time/Date: Thu Jun 12 15:13:51 2008
 Last Saved Time/Date: Sun Jul 20 01:28:03 2008
 Security: 0
```

d) Once you've done this, inspect the file metadata a second time using the `olemeta` utility. What interesting things did you find? Can you answer any of the customer's questions yet?

▼ **Hint**

If you're not familiar with `olemeta`, refer to its usage documentation:

```
analyst@forensics~$ olemeta -h
```

▼ **Solution**

```
analyst@forensics~$ olemeta m57biz.xls
olemeta 0.54 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues
--------------------------------------------------------------------------------
FILE: m57biz.xls

Properties from the SummaryInformation stream:
+---------------------+------------------------------+
|Property             |Value                         |
+---------------------+------------------------------+
|codepage             |1252                          |
|title                |                              |
|subject              |                              |
|author               |Alison Smith                  |
|keywords             |                              |
|comments             |                              |
|last_saved_by        |Jean User                     |
|create_time          |2008-06-12 15:13:51           |
|last_saved_time      |2008-07-20 01:28:03           |
...
```

Document all your findings, including the forensic artefacts you have extracted and analysed.

Exercise 4: Finding the Pivot

You've now reached a critical part of your investigation. You know that Jean possessed the document of concern, and you also know some additional information about the file, including its origin and when it was last updated, however you're still yet to find how the file was exfiltrated from Jean's computer. By Jean's own admission, at least you know that it was exfiltrated.

The aim of this exercise is to find a pivot to your next data point. This pivot is of critical importance to your forensic story and will tie the remainder of your investigation together. To complete this task, please take your time to peruse the data on Jean's computer. In doing so you'll build a profile of Jean's computer usage. What applications are installed? What applications do you think Jean uses? What personal data is stored?

> ▾ **Hint**
>
> Jean's personal files are stored in her user profile: `\Documents and Settings\Jean\My Documents\` , this could be a good place to begin.

Dynamically and/or statically inspect the filesystem contents using the techniques you've learned. Don't forget to look for deleted files.

> ▾ **Solution**
>
> Once you've found the file of interest, you can open it with Firefox:
>
> ```
> analyst@forensics~$ firefox alisonm57.html
> ```

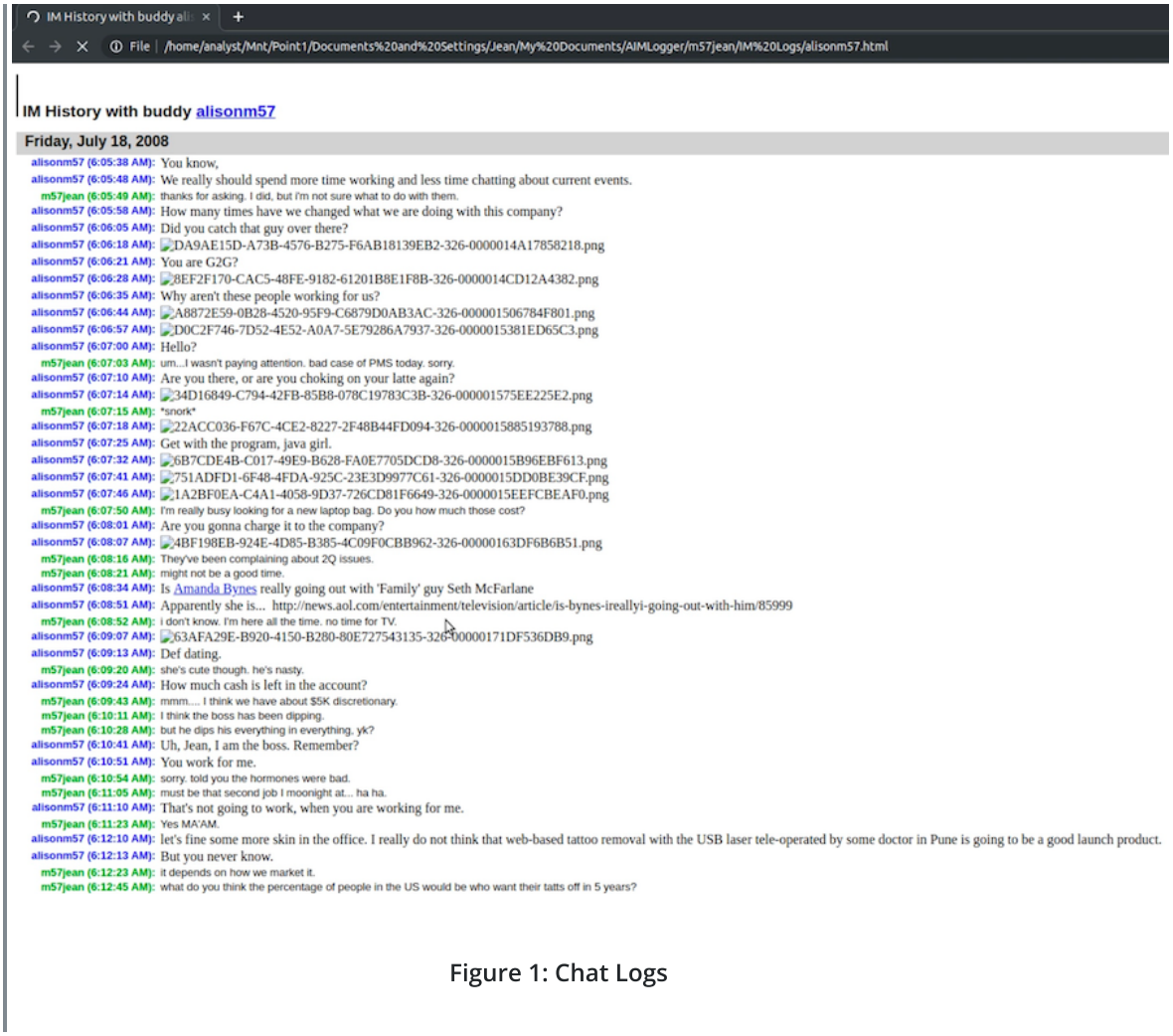

Figure 1: Chat Logs

Once you've found the right file, take note of the relevant time metadata, as this information is crucial to complete exercise five; it's your next pivot.

What conclusions can be drawn from the discussions held between the two parties? Document all your findings, including the forensic artefacts you've extracted and analysed.

Exercise 5: MFT Analysis

The first part of exercise five requires you to extract and process the filesystem's master file table (MFT).

a) Using the tsk utility `fls`, statically locate the MFT from Jean's computer. Once you've located the file, extract it to your analysis environment using the tsk utility `icat`.

▼ **Hint**

Using what you learned in earlier exercises, identify the correct inode of the MFT, then use `icat` to extract it to your analysis environment.

▼ **Solution**

```
analyst@forensics~$ fls out.bin | head -15
d/d 3519-144-6: Documents and Settings
r/r 4-128-4:    $AttrDef
r/r 8-128-2:    $BadClus
r/r 8-128-1:    $BadClus:$Bad
r/r 6-128-1:    $Bitmap
r/r 7-128-1:    $Boot
d/d 11-144-4:   $Extend
r/r 2-128-1:    $LogFile
r/r 0-128-1:    $MFT
r/r 1-128-1:    $MFTMirr
r/r 9-144-17:   $Secure:$SDH
r/r 9-144-16:   $Secure:$SII
r/r 9-128-18:   $Secure:$SDS
r/r 10-128-1:   $UpCase
r/r 3-128-3:    $Volume

analyst@forensics~$ icat out.bin 0-128-1 > MFT
analyst@forensics~$ ls -lah
total 21G
drwxr-xr-x 2 analyst analyst 4.0K Feb 16 01:48 .
drwxr-xr-x 3 analyst analyst 4.0K Feb 16 01:37 ..
-r--r--r-- 1 analyst analyst  10G Jul 10  2019 disk.bin
-rw-r--r-- 1 analyst analyst 285K Feb 16 01:36 m57biz.xls
-rw-r--r-- 1 analyst analyst  33M Feb 16 01:48 MFT
-rw-r--r-- 1 analyst analyst  10G Feb 16 01:27 out.bin
```

The MFT is typically near the start of a partition, so the `head` command is useful here.

b) In its raw state the MFT is not human readable. Using the `analyzeMFT.py` python utility, convert the MFT into a human readable comma separated values (CSV) file.

▼ **Hint**

If you're not familiar with `analyzeMFT.py`, refer to its usage documentation:

```
analyst@forensics~$ analyzeMFT.py -h
```

▼ **Solution**

Use `-f` to specify the filename to analyse, `-o` to specify your output filename, and `-w` to use \ instead of /:

```
analyst@forensics~$ analyzeMFT.py -f MFT -o MFT.csv -w
```

c) Once you've converted the MFT into a CSV file, open it in `Gnumeric`:

```
analyst@forensics~: gnumeric MFT.csv
```

Search for your data point from exercise four within the spreadsheet and pivot using temporal analysis of the surrounding filesystem events.

What can you deduce from the events that you observed? Do your deductions match your final conclusions from exercise four (they should!). Document all your findings, including the forensic artefacts you've extracted and analysed.

d) Of interest, Windows Prefetch files (introduced in Windows XP) are designed to speed up the application start-up process. Prefetch files contain the name of the executable, a Unicode list of DLLs used by that executable, a count of how many times the executable has been run, and a timestamp indicating the last time the program was run. Raw Windows Prefetch files are not human readable. However, you can use the `sccainfo` utility to convert Prefetch files into a human readable form.

▼ **Hint**

If you're not familiar with the `sccainfo` utility, refer to its usage documentation:

```
analyst@forensics~$ man 1 sccainfo
```

▼ **Solution**

According to our MFT research, there are some prefetch files in \Windows\Prefetch\

```
analyst@forensics~$ fls -rpd out.bin | grep Prefetch
r/- * 0:        WINDOWS/Prefetch/SHMGRATE.EXE-1BA69E68.pf
r/- * 0:        WINDOWS/Prefetch/OSA9.EXE-27CD7DB8.pf
r/- * 0:        WINDOWS/Prefetch/RUNDLL32.EXE-268BFF96.pf
```

If we want to analyse the existing prefetch, the easiest way is by mounting our binary image and running `sccainfo` on the files directly:

```
analyst@forensics~$ sccainfo /mnt/lab2/WINDOWS/Prefetch/EXCEL.EXE-1C75F8D6.pf
Windows Prefetch File (PF) information:
        Format version                  : 17
        Prefetch hash                   : 0x1c75f8d6
        Executable filename             : EXCEL.EXE
        Run count                       : 2
        Last run time:                  : Jul 20, 2008 01:27:40.718750000 UTC

Filenames:
```

```
        Number of filenames             : 59
...
        Filename: 16                    : \DEVICE\HARDDISKVOLUME1\PROGRAM
FILES\MICROSOFT OFFICE\OFFICE\EXCEL.EXE
...
        Filename: 39                    : \DEVICE\HARDDISKVOLUME1\DOCUME~1
\JEAN\LOCALS~1\TEMP\M57BIZ.XLS
...
Volumes:
        Number of volumes               : 1

Volume: 1 information:
        Device path                     : \DEVICE\HARDDISKVOLUME1
        Creation time                   : May 13, 2008 22:18:43.625000000 UTC
        Serial number                   : 0x744fc21f
```

See what other interesting programs were run around our time of interest.

Exercise 6: Exfiltration

You've now commenced the final lap of your investigation. You could possibly take a few guesses about what happened. However, a forensic analyst never takes a guess or assumes, so you need to find that smoking gun.

You know that Jean was using Microsoft Outlook just before she had the conversation with Alison. From both her discussion and interview you also know that she admitted to emailing the confidential document.

Of interest, Microsoft Outlook uses the Personal Storage Table (.pst) file format to store copies of messages, calendar events, and other items. Although a pst file can be split if it becomes excessively large, on most systems it's simply a single file.

a) Statically locate Jean's `pst` file using the tsk utility `fls`. Once you've located the file, extract it into your analysis environment using the tsk utility `icat`.

▾ **Hint**

Browse the filesystem using `fls -r out.bin inode`:

```
analyst@forensics~$ fls -r out.bin 3519-144-6
d/d 10222-144-6:        Administrator
d/d 3521-144-6: All Users
```

```
d/d 3520-144-7: Default User
d/d 17437-144-5:        Devon
d/d 16144-144-5:        Jean
d/d 10151-144-6:        LocalService
d/d 3368-144-6: NetworkService
```

▾ **Solution**

If we look in the inode of Jean's user profile for Outlook or .pst files, we discover Jean's pst:

```
analyst@forensics~$ fls -r out.bin 16144-144-5 | grep outlook
++++ r/r 17358-128-3:   outlook.pst
```

We can then export it with `icat`:

```
analyst@forensics~$ icat out.bin 17358-128-3 > outlook.pst
analyst@forensics~$ ls -lah outlook.pst
-rw-r--r-- 1 analyst analyst 2.3M Feb 16 01:57 outlook.pst
```

b) A `pst` file is not human readable in its raw state. Using the `pffinfo` and `pffexport` utilities, convert Jean's `pst` file into a human readable format.

▾ **Hint**

If you're not familiar with the `pffinfo` or `pffexport` utilities, refer to their usage documentation:

```
analyst@forensics~$ man 1 pffinfo
analyst@forensics~$ man 1 pffexport
```

▾ **Solution**

Run `pffexport` directly on the pst, with options if you so choose to make the output easier to manage:

```
analyst@forensics~$ pffexport outlook.pst
pffexport 20180714

Opening file.
Exporting items.
Exporting folder item 1 out of 5.
Exporting email item 1 out of 9.
Exporting recipient.
Exporting email item 2 out of 9.
Exporting recipient.
Exporting email item 3 out of 9.
Exporting recipient.
Exporting email item 4 out of 9.
Exporting recipient.
Exporting email item 5 out of 9.
```

```
Exporting recipient.
Exporting email item 6 out of 9.
Exporting recipient.
Exporting email item 7 out of 9.
Exporting recipient.
Exporting email item 8 out of 9.
Exporting recipient.
Exporting email item 9 out of 9.
Exporting recipient.
Exporting email item 1 out of 222.
Exporting attachment 1 out of 8.
...

analyst@forensics~$ cat outlook.pst.export/Top\ of\ Personal\ Folders/Sent\
Items/Message00018/Message.txt
I'm handling some business-trip-related stuff. any reservations you want me to
make on your behalf?

Ps What's this thing about Katie' Holmse's hands?

What are you doing?
```

c) Inspect Jean's extracted emails. What did you find? Can you now answer the customer's final questions? Did you find anything unexpected? Document all your findings and include all the forensic artefacts you've extracted and analysed.

If you've been consistently documenting everything so far, and haven't cut any corners, then turning your notes into a final report should be straightforward. Make sure your report tells the complete story (no omissions) and is entirely based on fact (no conjecture). The structure of the report should include a high-level executive summary which succinctly answers all the customers questions, furthermore a technical section should also be included that contains all your technical analytical findings. Be sure to include all the artefacts that you extracted, including their relevant information, to verify all your statements and claims. A complete forensic analysis should always be repeatable and deterministic, especially if legal proceedings are to follow.

Exercise 7: SuperTimeline (Bonus)

If you made it this far and still have time remaining, consider spending some time introducing yourself to `plaso`. `Plaso` is the Python-based back-end engine used by tools such as

`log2timeline` for the automatic creation of supertimelines. The goal of `log2timeline` (and `plaso`) is to provide a single tool that can parse various log files, and forensic artefacts from computers and related systems to produce a single correlated timeline. This timeline can then be easily analysed by forensic investigators and analysts, speeding up investigations by correlating the vast amount of information found on an average computer system. `Plaso` is intended to be used when creating supertimelines but also supports creating targeted timelines.

The number of file formats supported by `plaso` is vast and extensible, so if a particular artefact isn't supported you can easily add support yourself. `Plaso` is also more than just a development framework and contains command-line tools for analysts who just want to perform investigations and not do development. Spend some time reading the help for `log2timeline.py`, `pinfo.py`, and `psort.py`. Create a supertimeline using Jean's computer using the `log2timeline.py` utility and repeat your analysis.

▼ **Solution**

The simplest way to create a complete supertimeline is to use the `psteal` frontend:

(**NB: this command will take about an hour to complete with default RAM/CPU settings. You can safely ignore any 'deprecation' warnings.**)

```
analyst@forensics~$ psteal.py --source out.bin -o dynamic -w timeline-psteal.csv
plaso - psteal version 20190131

Storage file    : 20230216T020142-out.bin.plaso

Identifier              PID     Status          Memory          Events
Tags            Reports
Main                    2673    exporting       1.0 GiB         1769350 (0)     0
(0)           0 (0)
...
Processing completed.

********************************** Counter ***********************************
        Events processed : 1769354
     Events MACB grouped : 1735202
Duplicate events removed : 1166
------------------------------------------------------------------------------
Storage file is 20230216T020142-out.bin.plaso
```

The entire `timeline-psteal.csv` will likely be too large to open in Excel, so try filtering the artefacts down with the `log2timeline` and `plaso` filtering options.

Alternatively, it *might* open in `gnumeric`.

To create a more succinct, targeted timeline, use `log2timeline.py` and `plaso`.

▼ **Hint**

First, look at the `log2timeline.py` help file to get an idea of the commands and filters available to you:

```
analyst@forensics~$ log2timeline.py -h
usage: log2timeline.py [-h] [--troubles] [-V] [--artifact_definitions PATH]
[--custom_artifact_definitions PATH]
                       [--data PATH] [--artifact_filters ARTIFACT_FILTERS]
[--artifact_filters_file PATH]
                       [--preferred_year YEAR] [--process_archives]
[--skip_compressed_streams] [-f FILE_FILTER]
                       [--hasher_file_size_limit SIZE] [--hashers HASHER_LIST]
[--parsers PARSER_FILTER_EXPRESSION]
                       [--yara_rules PATH] [--partitions PARTITIONS] [--volumes
VOLUMES] [--language LANGUAGE_TAG]
                       [--no_extract_winevt_resources] [-z TIME_ZONE] [--no_vss]
[--vss_only] [--vss_stores VSS_STORES]
                       [--credential TYPE:DATA] [-d] [-q] [-u] [--info]
[--use_markdown] [--no_dependencies_check]
                       [--logfile FILENAME] [--status_view TYPE] [-t TEXT]
[--buffer_size BUFFER_SIZE]
                       [--queue_size QUEUE_SIZE] [--single_process]
[--process_memory_limit SIZE]
                       [--temporary_directory DIRECTORY] [--vfs_back_end TYPE]
[--worker_memory_limit SIZE]
                       [--worker_timeout MINUTES] [--workers WORKERS]
[--sigsegv_handler] [--profilers PROFILERS_LIST]
                       [--profiling_directory DIRECTORY] [--profiling_sample_rate
SAMPLE_RATE] [--storage_file PATH]
                       [--storage_format FORMAT] [--task_storage_format FORMAT]
                       [SOURCE]

log2timeline is a command line tool to extract events from individual
files, recursing a directory (e.g. mount point) or storage media
image or device.

More information can be gathered from here:
    https://plaso.readthedocs.io/en/latest/sources/user/Using-log2timeline.html

positional arguments:
  SOURCE                Path to a source device, file or directory. If the source
is a supported storage media device or
                        image file, archive file or a directory, the files within
are processed recursively.

options:
  -h, --help            Show this help message and exit.
  --troubles            Show troubleshooting information.
  -V, --version         Show the version information.
...
```

▾ **Hint**

It's recommended you use a filter file to reduce the number of files being parsed. An example `filter.txt` is included in the Lab 2 materials, but others can be found online (e.g. https://github.com/log2timeline/plaso/blob/main/data/filter_windows.txt).

▾ **Hint**

You can further limit the processing time by telling `log2timeline` not to hash files above a certain size, and by choosing a fast hashing algorithm, such as `MD5`.

▼ **Hint**

Finally, you should choose to parse only those artefacts which are of interest or that are relevant to your investigation. You can view the full list of artefact parsers by using the `list` command:

```
analyst@forensics~$ log2timeline.py --parsers list
WARNING: the version of plaso you are using is more than 6 months old. We
strongly recommend to update it.


********************************** Parsers ***********************************
                Name : Description
------------------------------------------------------------------------------
    android_app_usage : Parser for Android usage history (usage-history.xml)
                        files.
        apache_access : Parser for Apache access log (access.log) files.
          apt_history : Parser for Advanced Packaging Tool (APT) History log
                        files.
              asl_log : Parser for Apple System Log (ASL) files.
         bash_history : Parser for Bash history files.
              bencode : Parser for Bencoded files.
       binary_cookies : Parser for Safari Binary Cookie files.
              bsm_log : Parser for Basic Security Module (BSM) event auditing
                        files.
         chrome_cache : Parser for Google Chrome or Chromium Cache files.
   chrome_preferences : Parser for Google Chrome Preferences files.
             cups_ipp : Parser for CUPS IPP files.
  custom_destinations : Parser for Custom destinations jump list
                        (.customDestinations-ms) files.
                 czip : Parser for Compound ZIP files.
           dockerjson : Parser for Docker configuration and log JSON files.
                 dpkg : Parser for Debian package manager log (dpkg.log) files.
                esedb : Parser for Extensible Storage Engine (ESE) Database
                        File (EDB) format.
...
```

▼ **Solution (Alternate)**

All of this may then result in a `log2timeline` command like this:

(NB: You can safely ignore any 'deprecation' warnings.)

```
analyst@forensics~$ log2timeline.py -f filter.txt --hasher_file_size_limit 1
--hashers md5 --parsers
custom_destinations,lnk,mactime,mft,olecf,pe,recycle_bin,usnjrnl,winevt,winevtx,winjob,winreg
-d --logfile l2t.log --storage_file plaso.dmp disk.bin
```

Once this command eventually completes, you'll be left with a `plaso.dmp` file. This file can't be read in it's current form, so you need to convert it to something simpler, like a CSV. To do that, use `psort.py` :

```
analyst@forensics~$ psort.py --output_time_zone "UTC" -o L2tcsv plaso.dmp -w
timeline-plaso.csv
```

You can then open the resulting file in `gnumeric` or Excel. Consider filtering further around a time you've already identified malicious activity. More information can be found in the psort documentation: https://plaso.readthedocs.io/en/latest/sources/user/Using-psort.html