Lab 1 Report
Digital Forensics

Anton Fluch `anfl4215` Johan Bäckström `jobc5829`

September 25, 2017

Contents

1 Background

The evidence for the case where provided in a .zip file named Lab1.zip. This file produced the following hash sums:

Example 1: SHA256 and MD5 sum for Lab1.zip

```
sha256sum Lab1.zip
9c5d0bfbeccd75858426cfc84345e0a68687b0fc5662b715153aa88cefd60fba

md5sum Lab1.zip
c4a731672747131b8b457a77178ad386
```

When opening the zip file the following folders and files where present:

```
Lab1
├── Exercise1_Hashing
│   ├── erase
│   ├── erase.exe
│   ├── hello
│   ├── hello (2)
│   ├── hello (3)
│   ├── hello (4)
│   └── hello.exe
├── Exercise2_File_Identification
│   ├── 01
│   ├── 02
│   ├── 03
│   ├── 04
│   ├── 05
│   ├── 06
│   ├── 07
│   ├── 08
│   ├── 09
│   ├── 10
│   ├── 11
│   └── 12
├── Exercise3_Anti_Files_Forensics
│   ├── c.mp3
│   └── Suspicious_File
├── Exercise4_Acquisition
│   └── winxp.dvi
├── Exercise5_Cracking
│   ├── casssh.pdf
│   ├── ht.zip.tar.gpg
│   ├── Untitled 1.ods
│   ├── untitled.docx
│   ├── untitled_hash.txt
│   ├── wallet1.dat
│   └── wallet2
└── Exercise6_Steganography
    ├── c1l.png
    └── c2l.png
```

2 Exercise 1: Hashing

In order to maintain the chain of custody and to uniquely identify all files, the hash sum for SHA256 [1] and MD5[2] where calculated for all the files in the folder Exercise1_Hashing. In Kali Linux[3] it is possible to calculate the hash sum of a file using the bash shell [4]. For example, if you type the command:

Example 2: calculate sha256 sum of all files in folder

```
sha256sum *
```

It will calculate and display the hash sum for the SHA256 algorithm for all the files in the folder you are currently standing. This resulted in the following hash sums:

Example 3: Result of sha256 and md5sum

```
sha256sum *
1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56 erase
1316543942a8c6cd754855500cd37068edbbd8b31c4979d2825a4e799fed6102 erase.exe
fad878bd261840a4ea4a8277c546d4f46e79bbeb60b059cee41f8b50e28d0e88 hello
1316543942a8c6cd754855500cd37068edbbd8b31c4979d2825a4e799fed6102 hello (2)
60d13913155644883f130b85eb24d778314014c9479aedb5f6323bf38ad3a451 hello (3)
1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56 hello (4)
60d13913155644883f130b85eb24d778314014c9479aedb5f6323bf38ad3a451 hello.exe

md5sum *
da5c61e1edc0f18337e46418e48c1290 erase
cdc47d670159eef60916ca03a9d4a007 erase.exe
da5c61e1edc0f18337e46418e48c1290 hello
cdc47d670159eef60916ca03a9d4a007 hello (2)
cdc47d670159eef60916ca03a9d4a007 hello (3)
da5c61e1edc0f18337e46418e48c1290 hello (4)
cdc47d670159eef60916ca03a9d4a007 hello.exe
```

An efficient way for matching hash sums is also possible using the same command, but we need to provide an option to it. Using the '-c' option we can quickly check if a provided hash sum match with the file we are checking. First we need to create a new file with the hash sum for all the files in the folder:

Example 4: Save result in new file

```
sha256sum * > checksums.chk
```

This will create a new file named 'checksums.chk' which contains all the hash sums for the files in the folder. Then we run the command:

Example 5: Check if files il folder match with files in list

The output should be the following:

Example 6: Output from Example above

[1] https://en.wikipedia.org/wiki/SHA-2
[2] https://en.wikipedia.org/wiki/MD5
[3] https://www.kali.org/
[4] https://en.wikipedia.org/wiki/Bash_(Unix_shell)

```
erase: OK
erase.exe: OK
hello: OK
hello (2): OK
hello (3): OK
hello (4): OK
hello.exe: OK
```

Which indicates that all the files currently stored in 'checksums.chk' match with all the files in the folder. Now lets say that we have a specific file of interest which we know the hash sum of and we want to find out if the file is present on a computer. This can be achieced by using the following command:

Example 7: Command for finding file with hash sum

```
find . -type f -exec sha256sum {} + | grep '^SHA256SUM'
```

*Note that you need to replace 'SHA256SUM' with the actual hash value of the file

This will search through the specified folder recursively for correlating SHA256 sums. If we run the command:

Example 8: Finding files with hash sum

```
find . -type f -exec sha256sum {} + | grep
    '^1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56'
```

Which is the SHA256 sum of the file 'erase' mentioned above. We get the output:

Example 9: Result from above example

```
1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56 ./erase
1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56 ./hello (4)
```

This indicates that we found two files that both have the same SHA256 sum, 'erase' and 'hello (4)'.

This is a feature which should be considered as beneficial for a forensic examiner since it means that if you suspect that a file is present on a computer you can easily find it. Even though the file name is changed the hash sums will be identical.

2.1 Exercise 1.2: Comparison of Hashing Algorithms

In this exercise the execution time of the SHA256 and the MD5 algorithm will be compared. The file that is used to compare the times can be found at `http://ipv4.download.thinkbroadband.com:8080/1GB.zip` And should produce the following hash sums:

Example 10: Hash sums for file used in exercise

```
sha256sum
5674e59283d95efe8c88770515a9bbc80cbb77cb67602389fd91def26d26aed2

md5sum
286e80b3b7420263038ab06d76774043
```

Using the 'stat' command we can get more information about the file:

Example 11: Result from 'stat' command

```
stat 1GB.zip
     File: 1GB.zip
     Size: 1073741824    Blocks: 2097160  IO Block: 4096 regular file
   Device: 801h/2049d    Inode: 13369385  Links: 1
   Access: (0664/-rw-rw-r--) Uid: ( 1000/ fluchey) Gid: ( 1000/ fluchey)
   Access: 2017-09-21 11:56:19.516000051 +0200
   Modify: 2017-09-21 11:55:49.996055229 +0200
   Change: 2017-09-21 11:55:50.100055012 +0200
    Birth: -
```

If we want to measure the time it takes to compute tha hash sums we can use the command 'time'.

Example 12: Time taken for SHA256

```
time sha256sum 1GB.zip
5674e59283d95efe8c88770515a9bbc80cbb77cb67602389fd91def26d26aed2 1GB.zip

real    0m6,065s
user    0m5,968s
sys     0m0,100s
```

Example 13: Time taken for MD5

```
time md5sum 1GB.zip
286e80b3b7420263038ab06d76774043 1GB.zip

real    0m1,844s
user    0m1,732s
sys     0m0,108s
```

The 'time' command is described in more detail in the linux manual [5].

- 'real' The total time taken for the process to execute
- 'user' The amount of CPU time spent in user mode (Outside the kernel) within the process
- 'sys' The amount of CPU time spent in the kernel within the process

The SHA256 algorithm took a total of 6,065 seconds to run. The MD5 algorithm took a total of 1,844 seconds to run. This makes the MD5 algorithm 4,221 seconds faster.

[5] http://man7.org/linux/man-pages/man7/time.7.html

3 Exercise 2: File Headers

In the folder Exercise3_Anti_Files_Forensics a number of unidentified files where found. In order to make sure what kind of files they are we use three different tools for file identification and cross check their result.

3.1 file

In Kali Linux you can get information about files using the 'file' command. While standing in the 'Exercise2_File_Identification' folder and running the 'file *' command we get the following result:

Example 14: Result from 'file' command

```
file *
01: JPEG image data, JFIF standard 1.01, resolution (DPI), density 72x72,
    segment length 16, baseline, precision 8, 792x1024, frames 1
02: GIF image data, version 87a, 359 x 313
03: MS Windows 95 Internet shortcut text (URL=<http://www.dc3.mil/challenge/>),
    ASCII text, with CRLF line terminators
04: Zip archive data, at least v2.0 to extract
05: Zip archive data, at least v2.0 to extract
06: zlib compressed data
07: RPM v3.0 bin i386/x86_64
08: MS Windows HtmlHelp Data
09: Standard MIDI data (format 1) using 21 tracks at 1/240
10: ASCII text, with CRLF line terminators
11: Composite Document File V2 Document, Little Endian, Os: Windows, Version
    5.1, Code page: 1252, Title: , Subject: , Author: , Keywords: , Comments: ,
    Template: Normal.dot, Last Saved By: Kevin Allen, Revision Number: 37, Name
    of Creating Application: Microsoft Word 11.0, Total Editing Time:
    1d+16:02:00, Last Printed: Wed Sep 11 21:29:00 2002, Create Time/Date: Fri
    Jun 30 13:29:00 2000, Last Saved Time/Date: Wed Apr 2 19:07:00 2003, Number
    of Pages: 1, Number of Words: 10971, Number of Characters: 62539, Security:
    0
12: BitTorrent file
```

This gives us information of all the files in the folder.

3.2 HexEdit

On windows we can use the tool HexEdit[6] to get information about the files in hexadecimal form. We can then identify the hexadecimal header of the file and check on the website of Gary Kessler[7] for a matching file header. Doing this we get the following result:

[6] http://hexedit.com/

[7] http://www.garykessler.net/library/file_sigs.html

File	Header	Description
01	FF D8	Generic JPEGimage file
02	47 49 46 38 37 61	GIF87a (Graphics interchange format file)
03	-	No match on website using header. It is possible to see in HexEdit that it is some kind of Internet shortcut
04	50 4B 03 04	ZIP (PKZIP archive file)
05	50 4B 03 04	ZIP (PKZIP archive file)
06	78 01 63 60	No match on website using header
07	ED AB EE DB	RPM (Redhat Package manager file)
08	49 54 53 46	CHI, CHM (Microsoft Compiled HTML Help File)
09	4D 54 68 64	MID, MIDI (Musical Instrument Digital Interface (MIDI) sound file)
10	-	No match on website using header. It is possible to see in HexEdit that it is a README file for Microsoft File Checksum integrity Verifier V2.05
11	D0 CF 11 E0 A1 B1 1A E1	An Object Linking and Embedding (OLE) Compound File (CF) (i.e., OLECF) file format, known as Compound Binary File format by Microsoft, used by Microsoft Office 97-2003 applications (Word, Powerpoint, Excel, Wizard).
12	-	No match on website using header. In HexEdit you can see a description about a torrent file

3.3 TrId

TrIdNet[8] is another tool which presents it's findings in a GUI. Below are the result for all the files.

[8] `http://mark0.net/soft-tridnet-e.html`

Figure 1: File 01

	Match	Ext	Type	Pts
▶	38.1%	JPG	JFIF JPEG Bitmap	4003/3
	28.6%	JPG	JPEG Bitmap	3000/1
	23.8%	MP3	MP3 audio (ID3 v1.x tag)	2500/1/1
	9.5%	MP3	MP3 audio	1000/1

	Match	Ext	Type	Pts
▶	60.0%	GIF	GIF87a Bitmap	6001/2
	30.0%	GIF	GIF Bitmap (generic)	3000/1
	10.0%	BS/BIN	PrintFox (C64) bitmap	1000/1

Figure 2: File 02

	Match	Ext	Type	Pts
▶	91.7%	URL	Windows URL shortcut	11000/1/2
	8.3%	INI	Generic INI configuration	1000/1

Figure 3: File 03

	Match	Ext	Type	Pts
▶	100.0%	ZIP	ZIP compressed archive	4000/1

Figure 4: File 04

	Match	Ext	Type	Pts
▶	66.6%	XPI	Mozilla Firefox browser extension	8000/1/
	33.3%	ZIP	ZIP compressed archive	4000/1
	0.1%	CEL	Autodesk FLIC Image File (extensions: flc, fli, cel)	7/3

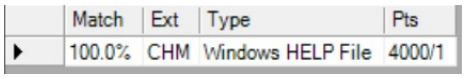
Figure 5: File 05

	Match	Ext	Type	Pts
▶	50.0%	DMG	Disk Image (Macintosh)	1000/1
	50.0%	XMI	XMill compressed XML	1000/1

Figure 6: File 06

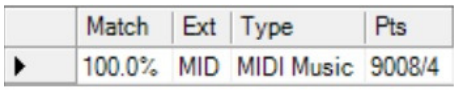
	Match	Ext	Type	Pts
▶	100.0%	RPM	RPM Package (generic)	4000/1

Figure 7: File 07



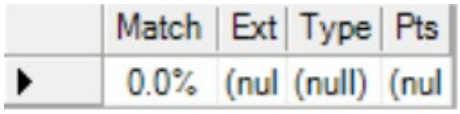
	Match	Ext	Type	Pts
▶	100.0%	CHM	Windows HELP File	4000/1

Figure 8: File 08



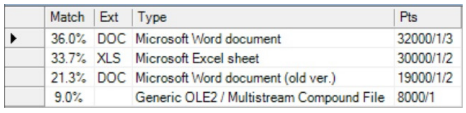
	Match	Ext	Type	Pts
▶	100.0%	MID	MIDI Music	9008/4

Figure 9: File 09



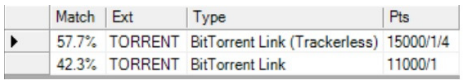
	Match	Ext	Type	Pts
▶	0.0%	(nul	(null)	(nul

Figure 10: File 10



	Match	Ext	Type	Pts
▶	36.0%	DOC	Microsoft Word document	32000/1/3
	33.7%	XLS	Microsoft Excel sheet	30000/1/2
	21.3%	DOC	Microsoft Word document (old ver.)	19000/1/2
	9.0%		Generic OLE2 / Multistream Compound File	8000/1

Figure 11: File 11



	Match	Ext	Type	Pts
▶	57.7%	TORRENT	BitTorrent Link (Trackerless)	15000/1/4
	42.3%	TORRENT	BitTorrent Link	11000/1

Figure 12: File 12

3.4 Conclusion

After identifying the file types with the three tools mentioned above, the files where opened with the corresponding program for further examination. The following table presents the conclusion of the findings:

File	Description
01	JPEG image of a hangar filled with airplanes
02	GIF image of a generator or a motor
03	A windows shortcut for URL address containing `http://www.dc3.mil/challenge`
04	ZIP archive containing the tool EXIF.exe and documentation
05	ZIP archive containing Chrome plugin and install files
06	ZLIb archive - unknown contents
07	Red hat package manager archive
08	Nvidia control panel help file
09	MIDI file with the song Carmina Burana - O Fortuna
10	README file about Microsoft Check File Integrity Verifier
11	OLE file with a README about the tool Robocopy
12	.torrent file for Ubuntu ISO (AMD64 version)

4 Exercise 3: Anti Files Forensics

Inside the folder Exercise3_Anti_Files_Forensics. Two files named 'c.mp3' and 'Suspicious_file' where present. The hash sums for both files are listed below:

Example 15: SHA256 and MD5 sum of files in folder

```
sha256sum *
83a15326cf9066a36defbe4f8a0633ec16867999c5910257807493ce250a3548 c.mp3
cec6534e8ddc4f5f9e9b2a0cedb438a8419a5ffd08ecfe059467630f624d5b1a Suspicious_File

md5sum *
670a8c0db494ced4882b44b27dbd6af2 c.mp3
63017bb2a213fa440191b204929ab0f7 Suspicious_File
```

More information about the files could be obtained by using the file command:

Example 16: Result from 'file' command

```
c.mp3: Audio file with ID3 version 2.255.216, unsynchronized frames, extended
    header, experimental, footer present

Suspicious_File: Composite Document File V2 Document, Cannot read section info
```

4.1 Foremost

Using the Linux tool foremost[9] it was discovered that the files contained other content than the file ending indicated.

4.1.1 c.mp3

Using foremost on the file 'c.mp3' one additional .jpg file were found.

Example 17: foremost file 'c'

```
foremost c.mp3 -v -o c

Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Sun Sep 24 16:11:49 2017
Invocation: foremost c.mp3 -v -o c
Output directory:
    /home/fluchey/Documents/Skola/DIFO/Lab1/Exercise3_Anti_Files_Forensics/c
Configuration file: /etc/foremost.conf
Processing: c.mp3
|------------------------------------------------------------------
File: c.mp3
Start: Sun Sep 24 16:11:49 2017
Length: 18 KB (19332 bytes)

Num     Name (bs=512)       Size     File Offset    Comment

0:      00000000.jpg        18 KB             3
*|
Finish: Sun Sep 24 16:11:49 2017
```

[9] https://en.wikipedia.org/wiki/Foremost_(software)

```
1 FILES EXTRACTED

jpg:= 1 # Here we can see that one additional .jpg file were extracted
-----------------------------------------------------------------

Foremost finished at Sun Sep 24 16:11:49 2017
```

Further examination of the extracted file showed that it was an image of the Actress Keira Knightley[10] and the file was named '00000000.jpg'.

Figure 13: Extracted picture from 'c.mp3'

4.1.2 Suspicious_File

Using foremost on the file 'Suspicious_File' one additional .ole[11] file were found

Example 18: foremost file 'Suspicious_File'

```
foremost Suspicious_File -v -o sus
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Sun Sep 24 16:19:26 2017
Invocation: foremost Suspicious_File -v -o sus
Output directory:
    /home/fluchey/Documents/Skola/DIFO/Lab1/Exercise3_Anti_Files_Forensics/sus
Configuration file: /etc/foremost.conf
Processing: Suspicious_File
|------------------------------------------------------------------
File: Suspicious_File
```

[10] http://www.imdb.com/name/nm0461136/
[11] https://en.wikipedia.org/wiki/Object_Linking_and_Embedding

```
Start: Sun Sep 24 16:19:26 2017
Length: 1 MB (1304576 bytes)

Num     Name (bs=512)       Size     File Offset    Comment

0:      00000000.ole        12 KB             0
*|
Finish: Sun Sep 24 16:19:26 2017

1 FILES EXTRACTED

ole:= 1 # Here we can see that on .ole file were extracted
------------------------------------------------------------------

Foremost finished at Sun Sep 24 16:19:26 2017
```

The extracted .ole file failed to open in any recommended software. Both 'Suspicious_File' and the extracted .ole file were submitted to malware scanning sites VirusTotal[12] and Cryptam[13]

[12] https://www.virustotal.com/#/home/upload
[13] https://www.cryptam.com/

5 Exercise 4: Acquisition

6 Exercise 5:

7 Exercise 6 - Hashing