Lab 1 Report
Digital Forensics

Anton Fluch `anfl4215`   Johan Bäckström `jobc5829`

October 2, 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
```

For the file calc.exe, the hash sum was calculated using md5deep64.exe with the following command:

Example 4: Command for md5deep64.exe

```
  C:\Users\cs2lab\Desktop\Forensic_tools\md5deep-4.4>findstr /i
      10e4a1d2132ccb5c6759f038cdb6f3c9 "c:\users\cs2lab\Desktop\Shared
      Folder\NSRL_Hash_sums_Database.txt"
```

The hash sum was searched for within the file NSRL_hash_sums_Database.txt

[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)

Example 5: Result for findstr

```
  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,19417,"358",""

  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,19423,"358","D"

  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,19487,"358","D"

  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,22270,"358",""

  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,22273,"358",""

  "42D36EEB2140441B48287B7CD30B38105986D68F","10E4A1D2132CCB5C6759F038CDB6F3C9",
  "967E5DDE","calc.exe",918528,23807,"358",""
```

This means that we found the md5 hash sum for calc.exe within that file.

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 6: 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 7: Check if files in folder match with files in list

```
sha256sum -c checksums.chk
```

The output should be the following:

Example 8: Output from Example above

```
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 9: 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 10: 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 11: 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 12: Hash sums for file used in exercise

```
sha256sum
5674e59283d95efe8c88770515a9bbc80cbb77cb67602389fd91def26d26aed2

md5sum
286e80b3b7420263038ab06d76774043
```

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

Example 13: 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 14: Time taken for SHA256

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

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

Example 15: 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 16: 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.

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

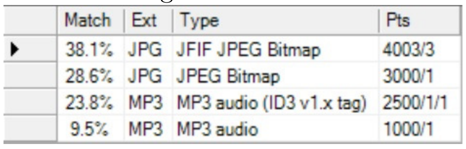
File	Header	Description
01	FF D8	Generic JPEGimage file
02	47 49 46 38 37 61	GIF87a (Graphics interchange format file)
03	-	No match on website using header. It is possible to see in HexEdit that it is some kind of Internet shortcut
04	50 4B 03 04	ZIP (PKZIP archive file)
05	50 4B 03 04	ZIP (PKZIP archive 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

3.2 TrId

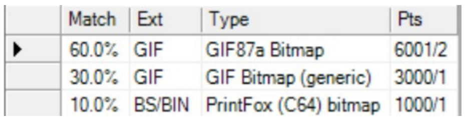
TrIdNet[8] is another tool which presents it's findings in a GUI. Below are the result for all the files.

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

Figure 1: File 01

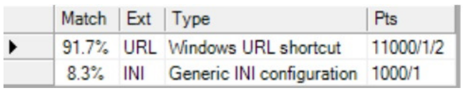


	Match	Ext	Type	Pts
▶	38.1%	JPG	JFIF JPEG Bitmap	4003/3
	28.6%	JPG	JPEG Bitmap	3000/1
	23.8%	MP3	MP3 audio (ID3 v1.x tag)	2500/1/1
	9.5%	MP3	MP3 audio	1000/1



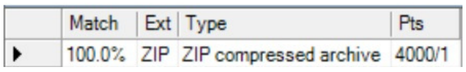
	Match	Ext	Type	Pts
▶	60.0%	GIF	GIF87a Bitmap	6001/2
	30.0%	GIF	GIF Bitmap (generic)	3000/1
	10.0%	BS/BIN	PrintFox (C64) bitmap	1000/1

Figure 2: File 02



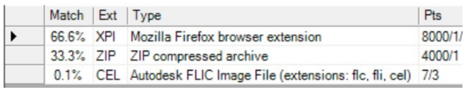
	Match	Ext	Type	Pts
▶	91.7%	URL	Windows URL shortcut	11000/1/2
	8.3%	INI	Generic INI configuration	1000/1

Figure 3: File 03



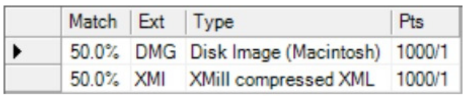
	Match	Ext	Type	Pts
▶	100.0%	ZIP	ZIP compressed archive	4000/1

Figure 4: File 04



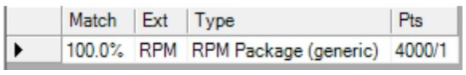
	Match	Ext	Type	Pts
▶	66.6%	XPI	Mozilla Firefox browser extension	8000/1/
	33.3%	ZIP	ZIP compressed archive	4000/1
	0.1%	CEL	Autodesk FLIC Image File (extensions: flc, fli, cel)	7/3

Figure 5: File 05



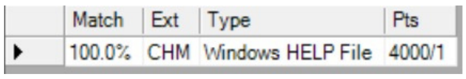
	Match	Ext	Type	Pts
▶	50.0%	DMG	Disk Image (Macintosh)	1000/1
	50.0%	XMI	XMill compressed XML	1000/1

Figure 6: File 06



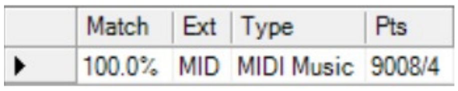
	Match	Ext	Type	Pts
▶	100.0%	RPM	RPM Package (generic)	4000/1

Figure 7: File 07



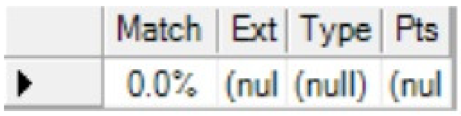
	Match	Ext	Type	Pts
▶	100.0%	CHM	Windows HELP File	4000/1

Figure 8: File 08



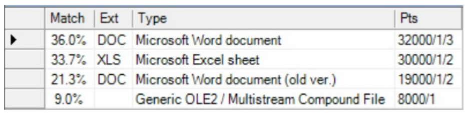
	Match	Ext	Type	Pts
▶	100.0%	MID	MIDI Music	9008/4

Figure 9: File 09



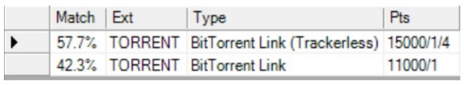
	Match	Ext	Type	Pts
▶	0.0%	(nul	(null)	(nul

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

3.3 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 17: 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 18: 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 19: 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'.

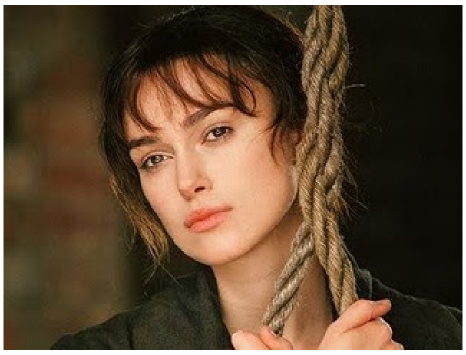


Figure 13: Extracted picture from 'c.mp3'

4.1.2 Suspicious_File

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

Example 20: foremost file 'Suspicious_File'

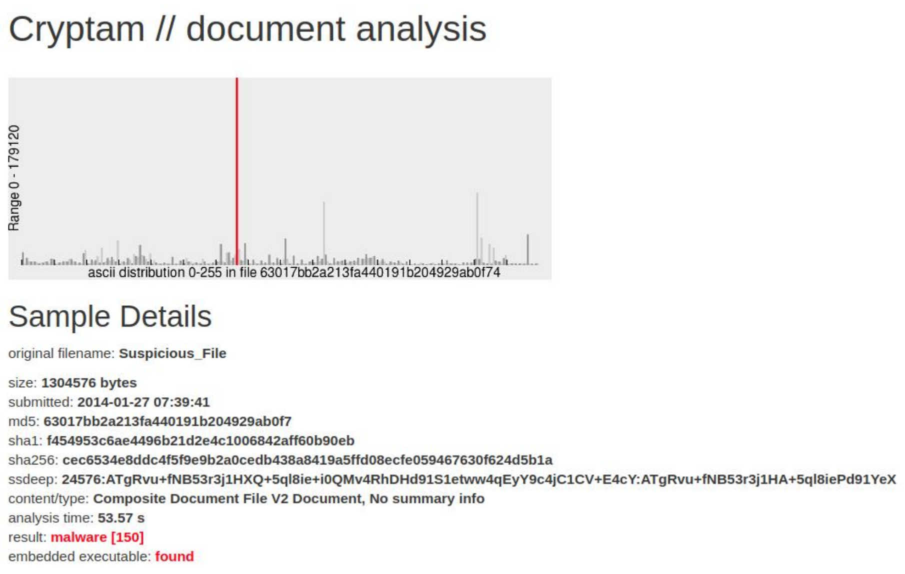
```
foremost Suspicious_File -v -o sus
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

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
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 one .ole file were extracted
```

[10] http://www.imdb.com/name/nm0461136/

[11] https://en.wikipedia.org/wiki/Object_Linking_and_Embedding

```
------------------------------------------------------------------
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 site Cryptam[12]. Cryptam discovered that 'Suspicious_File' contained an embedded executable.



Range 0 - 179120
ascii distribution 0-255 in file 63017bb2a213fa440191b204929ab0f74

Cryptam // document analysis

Sample Details

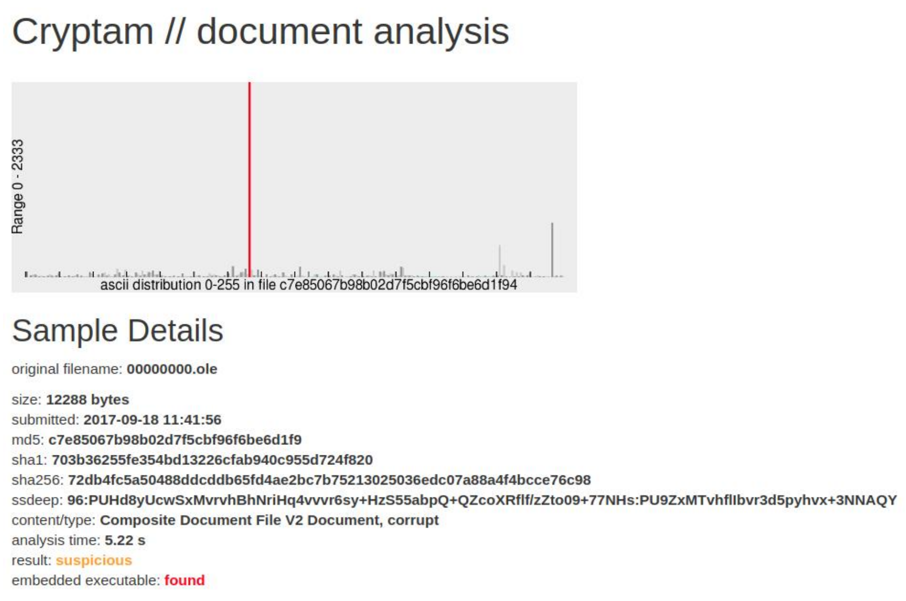
original filename: **Suspicious_File**

size: **1304576 bytes**
submitted: **2014-01-27 07:39:41**
md5: **63017bb2a213fa440191b204929ab0f7**
sha1: **f454953c6ae4496b21d2e4c1006842aff60b90eb**
sha256: **cec6534e8ddc4f5f9e9b2a0cedb438a8419a5ffd08ecfe059467630f624d5b1a**
ssdeep: **24576:ATgRvu+fNB53r3j1HXQ+5ql8ie+i0QMv4RhDHd91S1etww4qEyY9c4jC1CV+E4cY:ATgRvu+fNB53r3j1HA+5ql8iePd91YeX**
content/type: **Composite Document File V2 Document, No summary info**
analysis time: **53.57 s**
result: **malware [150]**
embedded executable: **found**

Figure 14: Cryptam result of 'Suspicious_File'

And it found the .ole file to be suspicious

[12]https://www.cryptam.com/



Cryptam // document analysis

Range 0 - 2333
ascii distribution 0-255 in file c7e85067b98b02d7f5cbf96f6be6d1f94

Sample Details

original filename: **00000000.ole**

size: **12288 bytes**
submitted: **2017-09-18 11:41:56**
md5: **c7e85067b98b02d7f5cbf96f6be6d1f9**
sha1: **703b36255fe354bd13226cfab940c955d724f820**
sha256: **72db4fc5a50488ddcddb65fd4ae2bc7b75213025036edc07a88a4f4bcce76c98**
ssdeep: **96:PUHd8yUcwSxMvrvhBhNriHq4vvvr6sy+HzS55abpQ+QZcoXRflf/zZto09+77NHs:PU9ZxMTvhflIbvr3d5pyhvx+3NNAQY**
content/type: **Composite Document File V2 Document, corrupt**
analysis time: **5.22 s**
result: **suspicious**
embedded executable: **found**

Figure 15: Cryptam result of '00000000.ole'

5 Exercise 4: Acquisition

See separate file named 'DIFO2017_Group29_Lab1_Report_Template_B_Assignment4'

6 Exercise 5: Cracking

The folder Exercise5_Cracking contained 7 files

Example 21: Content of Exercise5_Cracking folder

```
sha256sum *
126b46eb0c3891f86f38af95b810ed083f242e5aaaa5bbd84caf9f1ee28af6a3 Untitled 1.ods
a5ba8562717a9c128bdb87d3b0b327cea7bb24f33fd599d5de8fc6d41c174d02 casssh.pdf
2fe1cd6a80f6609efbc4fc142ce552ef46155f0fe1b0b765f0b12ea7fd5d8f8b ht.zip.tar.gpg
c28739a4507ca3f39ac0cf1a06de6f58d410b3a1fb8724268264d1cbe67e8112 untitled.docx
c0b382ff5916cae9fb773193b6d225329724982a8f53ed9ba9bbbef001ba45f4
    untitled_hash.txt
6c073f77c40ca84a1ada73452633b58359d055e37ae75dcec0f663538db859a5 wallet1.dat
cb17a6d50d1992fb77131c3026e375404a6136445ddfa605a8c5e38d24b890a1 wallet2

md5sum *
0a4532f87c41c31f1b716cd21ea8fa51 Untitled 1.ods
f77f0c2ea19035ace1d47b266245984d casssh.pdf
159ef8f471caef32db49dc9b331e10a5 ht.zip.tar.gpg
3a9b988f1496b2598cf08023603bcf3c untitled.docx
94ad3d937cb04ea0772bfbd0e20564bb untitled_hash.txt
fbf3b26fcf8e9fe4ec2e70175255e36c wallet1.dat
91abbea64a2a922c6512232ca8d68fa1 wallet2
```

Using Password Recovery Toolkit [13], which is used to recover passwords.
The first file to be tried was **casssh.pdf**.

Using a big compilation of Swedish words[14] provided by the CS2Labs, as well as rules to append digits after and prepend digits before the words from the wordlist, the file was succesfully cracked.

The second file to be tried was **untitled.docx**. PRTK [15] was set to try the same rules that worked for casssh.pdf, i.e. using the words from the dictionary as-is and append up to 2 digits and prepend up to 2 digits. However, this attack tried very few passwords per second, (averaging 10-50) so this attack was aborted.

After this failed attempt, Hashcat[16] was used to try to crack the file untitled_hash.txt, which is the hash sum extracted from the word file.

Using a hybrid mask attack, to append two digits and prepend two digits together with the same dictionary as before, as seen below:

```
.\hashcat64.exe -a 7 -w 3 -m 9600 --status "C:\Users\Johan\Goo
gle Drive\DSV 5 HT-17\DIFO\Lab files\Lab1 original
    files\Exercise5_Cracking\untitled_hash.txt" ?d?d "C:\Users\Johan\Google
    Drive\DSV
5 HT-17\DIFO\Lab files\swedish_dict\SVENSK-ORDLISTA-COMPILATION.txt"
```

This attack failed to find any password.

```
.\hashcat64.exe -a 6 -w 3 -m 9600 --status "C:\Users\Johan\Goo
```

[13] http://accessdata.com/product-download/password-recovery-toolkit-prtk-version-8.1.0

[14] SVENSK-ORDLISTA-COMPILATION.txt

[15] http://accessdata.com/product-download/password-recovery-toolkit-prtk-version-8.1.0

[16] hashcat 3.6.0

```
gle Drive\DSV 5 HT-17\DIFO\Lab files\Lab1 original
     files\Exercise5_Cracking\untitled_hash.txt" "C:\Users\Johan\Google
     Drive\DSV 5 HT
-17\DIFO\Lab files\swedish_dict\SVENSK-ORDLISTA-COMPILATION.txt" ?d?d
```

This attack also failed to find any password.

Using hashcat a dictionary attack using top 95 thousand probable passwords[17] were also tried without success.

The other files were also tried in PRTK but failed to be cracked. According to PRTK with the specified rule sets and dictionaries, the cracking process would for some files take multiple days to be completed and there was no time for completing that in this lab.
The password cracking computers that were supposed to be available to us were unavailable for the entirety of the lab. We could not log on without kicking other users off of the system.

In the list below, the files and any recovered passwords are listed.

File	**Password**
Untitled 1.ods	Not found
casssh.pdf	01frid
ht.zip.tar.gpg	Not found
untitled.docx	Not found
untitled_hash.txt	Not found
wallet1.dat	Not found
wallet2	Not found

[17] `https://github.com/berzerk0/Probable-Wordlists`

7 Exercise 6 - Steganography

in the folder Exercise6_Steganography two files named 'c1l.png' and 'c2l.png' were present.

Example 22: SHA256 and MD5 sum of files in Exercise6_Steganography

```
sha256sum *
d8ad7abb90ee967108f37d3b702016827219692e25fc840fb7e737ba0b7eab00 c1l.png
822e27042277d9588d699adcf1f3da7428cad7de9506eda7ae3644f858574e1f c2l.png

md5sum *
601450fd443b42f4ece0e3f001ed73b3 c1l.png
b914cc2043a6d5b31ff4bb6f5f1291fc c2l.png
```

The file ending '.png' indicate that the files are images. Opening them in an image viewing program confirms that they are in fact images.

Figure 16: Image of 'c1l.png'

Figure 17: Image of 'c2l.png'

The files were opened in the hex editor Bless[18] to visually examine them. Nothing out of the ordinary was found for either file. The files were also opened in Pngcheck[19] and the following was reported:

[18]http://www.forensicswiki.org/wiki/Bless
[19]http://www.libpng.org/pub/png/apps/pngcheck.html

Example 23: c1l.png

```
No errors detected in c1l.png (7 chunks, 92.7% compression).
```

Example 24: c2l.png

```
No errors detected in c2l.png (3 chunks, 92.7% compression).
```

Opening the files in the program Stepic[20] with the command:

Example 25: stepic command

```
stepic -d -i ~/Desktop/Lab1/Exercise6_Steganography/c1l.png -o c1lout

stepic -d -i ~/Desktop/Lab1/Exercise6_Steganography/c2l.png -o c2lout
```

Reveals the following information:

Output for c1l.png, contains:

The text \FF (while viewed in Linux/Mac OS)
xFF (while viewed through Notepad++ [21] in Windows)

Different character sets were tried to see if the text could be understood.

Output for c2l.png, contains:

```
http://xdsa5xcrrrxxxolc.onion/
```

Which is an address for the TOR network, an address for an online pharmacy.
To find out more information about the two files and to find if there was any more hidden messages, a comparison of the two files were performed with the command:

```
compare c1l.png c2l.png compareresult.png
```

Which produces a new image containing the result of the the image comparation. I.e. the resulting differences between the two files.
The red portion shows a detected difference between the two files. There are differences between the two files and it is contained in the red highlighted portion in the above image.
The files were also examined by ExifTool[22]. The output from ExifTool differed and som information were not present in 'c2l.png'

Example 26: ExifTool output for 'c1l.png'

```
exiftool c1l.png
ExifTool Version Number      : 10.60
File Name                    : c1l.png
```

[20] http://domnit.org/stepic/doc/
[21] https://notepad-plus-plus.org/
[22] https://en.wikipedia.org/wiki/ExifTool

Figure 18: Image of 'compareresult.png'

```
Directory                       : .
File Size                       : 18 kB
File Modification Date/Time     : 2017:08:09 16:36:22+02:00
File Access Date/Time           : 2017:09:28 10:34:39+02:00
File Inode Change Date/Time     : 2017:09:15 12:24:53+02:00
File Permissions                : rwxr-xr-x
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 403
Image Height                    : 157
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Significant Bits                : 8 8 8 8       # // These rows are not
Pixels Per Unit X               : 7500          # // on the
Pixels Per Unit Y               : 7500          # // output for
Pixel Units                     : meters        # // c2l.png
Image Size                      : 403x157
Megapixels                      : 0.063
```

Example 27: ExifTool output for 'c2l.png'

```
exiftool c2l.png
ExifTool Version Number         : 10.60
File Name                       : c2l.png
Directory                       : .
File Size                       : 18 kB
File Modification Date/Time     : 2017:08:09 16:40:16+02:00
File Access Date/Time           : 2017:09:28 10:36:15+02:00
File Inode Change Date/Time     : 2017:09:15 12:24:53+02:00
File Permissions                : rwxr-xr-x
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 403
Image Height                    : 157
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
```

```
Interlace                     : Noninterlaced
Image Size                    : 403x157
Megapixels                    : 0.063
```

The files were further examined by the tool StegSolve [23] but no further findings were found.

This leads to the conclusion that cl1.png is the original and that c2l.png is the only image containing a steganographic hidden message.

[23] http://www.caesum.com/handbook/Stegsolve.jar