Lab 1 Report
Digital Forensics

Anton Fluch `anfl4215` Johan Bäckström `jobc5829`

September 21, 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:

SHA256
9c5d0bfbeccd75858426cfc84345e0a68687b0fc5662b715153aa88cefd60fba

MD5
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:

```
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:

```
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:

```
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:

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

The output should be the following:

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

[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)

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:

```
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:

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

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

```
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:

```
sha256sum
5674e59283d95efe8c88770515a9bbc80cbb77cb67602389fd91def26d26aed2

md5sum
286e80b3b7420263038ab06d76774043
```

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

```
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'.

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

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

```
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

4 Exercise 3: Anti Files Forensics

5 Exercise 4: Acquisition

6 Exercise 5:

7 Exercise 6 - Hashing