

Exercise 1 - Hashing

Objective: Our objective for this exercise was to find the hash values of all four files and to verify the hash values match from a text file that contains all the hash values of the four files.

Tools: md5deep64.exe version 4.4, sha256deep64.exe version 4.4 and whirlpooldeep64.exe version 64.

- **What files (and/or folders) did you hash?**

'hello', 'hello (2)', 'hello (3)' and 'hello (4)' files were all hashed using md5deep64.exe, sha256deep64.exe and whirlpooldeep64.exe.

The commands used were:

i) This is using md5deep64:

```
> C:\...path...\md5deep64.exe hello
da5c61e1edc0f18337e46418e48c1290 "hello"

> C:\...path...\md5deep64.exe "hello (2)"
cdc47d670159eef60916ca03a9d4a007 "hello (2)"

> C:\...path...\md5deep64.exe "hello (3)"
cdc47d670159eef60916ca03a9d4a007 "hello (3)"

> C:\...path...\md5deep64.exe "hello (4)"
da5c61e1edc0f18337e46418e48c1290 "hello"
```

ii) This is using sha256deep64:

```
> C:\...path...\sha256deep64.exe hello
fad878bd261840a4ea4a8277c546d4f46e79bbeb60b059cee41f8b50e28d0e88 C:\...path...\hello

> C:\...path...\sha256deep64.exe "hello (2)"
1316543942a8c6cd754855500cd37068edb8b31c4979d2825a4e799fed6102 "C:\...path...\hello (2)"

> C:\...path...\sha256deep64.exe "hello (3)"
60d13913155644883f130b85eb24d778314014c9479aedb5f6323bf38ad3a451 "C:\...path...\hello (3)"

> C:\...path...\sha256deep64.exe "hello (4)"
1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56 "C:\...path...\hello (4)"
```

iii) This is using whirlpooldeep64:

```
> C:\...path...\whirlpooldeep64.exe "C:\...path...\hello"
1f4388f4a81a6cfacde955cf5fd84c4e76f12876db3356e2a84efda91f8c44407b3626b770d9752f9b0aa05927e
7fb7c66e07ea96ea47ece2ca78a40cedb9d7e

> C:\...path...\whirlpooldeep64.exe "C:\...path...\hello (2)"
aae0a840704962b3026cf5b2058aa1a3d7752e6d562e0a843ce0abf7107666fb475ac45df08587c468f7754847
f2be4cebd1172dedfebfaa6527c8da2b1bc364
```

```
> C:\...path...\whirlpooldeep64.exe "C:\...path...\hello (3)"

959593fe62721c9d058831eaa742f103d1b853b43a81a94dd30cc89a6dafb6926b64bdd7b0ef16a4c566601f8
1af95988fb952720482562a6269728ef5f5d5fb

> C:\...path...\whirlpooldeep64.exe "C:\...path...\hello (4)"

4324731f2340b0d5ce741f566abd7ebf8a1c18a15e4835bd8c25bdc974ec275a6e088982422456060bf145d70
13cd0aa8ac1b62ee3f8149bfd6490e6a5a95efa
```

- **Which algorithms did you choose?**

Initially, we only used md5 algorithm and we noticed using md5, the 'hello' with 'hello (4)' files would have the same hash values where 'hello (2)' with 'hello (3)' files would also have the same hash values. Because of this finding, we decided to use sha256 and whirlpool algorithms to ensure and confirm if these files truly would have the same hash values. While using sha256 and whirlpool algorithms, we discovered the hash values were not the same for all four files.

The algorithms chosen for hashing 'hello', 'hello(2)', 'hello(3)' and 'hello(4)' were sha256, whirlpool and md5. The reason algorithm sha256 and whirlpool generated different hash values were: sha256 digested input message length up to 2^{64} bits in length and whirlpool digested input message length up to 2^{256} which resulted into different hash values for all four files. Whirlpool generated 512 bits length and sha256 generated 256 bits length.

- **Suppose you would need to calculate the hash sums of several files and folders, how could you do that using the tools and resources provided? Provide a description and exact commands.**

According to the <http://md5deep.sourceforge.net/>¹, '-r' flag can be used to hash all files within a folder recursively. This '-r' option doesn't hash the folder but only the files inside a folder.

Command used:

```
> C:\..input path...\whirlpooldeep64.exe -t -r C:\..input folder path > output_whirlpool.txt
```

Results from Output_test.txt:

```
1f4388f4a81a6cfacde955cf5fd84c4e76f12876db3356e2a84efda91f8c44407b3626b770d9752f9b0aa0592
7e7fb7c66e07ea96ea47ece2ca78a40cedb9d7e 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello
```

```
959593fe62721c9d058831eaa742f103d1b853b43a81a94dd30cc89a6dafb6926b64bdd7b0ef16a4c56660
1f81af95988fb952720482562a6269728ef5f5d5fb 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (3)
```

```
aae0a840704962b3026cf5b2058aa1a3d7752e6d562e0a843ce0abf7107666fb475ac45df08587c468f7754
847f2be4cebd1172dedfebfaa6527c8da2b1bc364 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (2)
```

```
4324731f2340b0d5ce741f566abd7ebf8a1c18a15e4835bd8c25bdc974ec275a6e088982422456060bf145
d7013cd0aa8ac1b62ee3f8149bfd6490e6a5a95efa 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (4)
```

```
> C:\..input path...\md5deep64.exe -t -r C:\..input folder path > output_md5.txt
```

Results from Output_test.txt:

```
da5c61e1edc0f18337e46418e48c1290 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello
```

¹ the <http://md5deep.sourceforge.net/>

```

cdc47d670159eef60916ca03a9d4a007 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (2)

cdc47d670159eef60916ca03a9d4a007 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (3)

da5c61e1edc0f18337e46418e48c1290 2017:09:14:12:27:34
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (4)

> C:\..\input path... \shar256deep64.exe -r C:\..\input folder path > output_md5.txt

Results from Output_test.txt:
fad878bd261840a4ea4a8277c546d4f46e79bbeb60b059cee41f8b50e28d0e88
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello

1316543942a8c6cd754855500cd37068edb8b31c4979d2825a4e799fed6102
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (2)

60d13913155644883f130b85eb24d778314014c9479aedb5f6323bf38ad3a451
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (3)

1c4ff4e490b15b2b214f26c5654decccbcbea9eb900f88649dc7b1e42341be56
C:\Users\cs2lab\Desktop\Shared_Folder\DiFo2017\Lab1\Exercise1_Hashing\test\hello (4)

```

- Come up with an efficient way to match hash sums – so that you can automatically identify files that might (or might not be) of interest. Describe how you did this and which exact programs you used, and reflect upon why this might be useful for a forensic examiner in their work life.**

The *findstr* command could be used to match and verify hash values of a file but this method was not as efficient as using the *md5deep64.exe*, *sha256deep64.exe* and *whirlpooldeep64.exe* with the ‘-m’ option directly. We also used ‘-r’ to recursively hash all files in a folder. The reason was *findstr* was not a tool to generate hash values. It would only search string values and compared the string to a file that contained the particular searched string. With *md5deep64.exe*, *sha256deep64.exe* and *whirlpooldeep64.exe*, they would generate hash values and would match the hash values of the files to the file that would contain the hash values to be verified.

Example commands used were:

```

> C:\..\input path... \md5deep64.exe -rm C:\..\input folder path...\ 
> C:\..\input path... \sha256deep64.exe -rm C:\..\input folder path...\ 
> C:\..\input path... \whirlpooldeep64.exe -rm C:\..\input folder path...\ 

```

The test files we used are in our test folder such as:

install61.iso (214428 KB), test1.txt (1KB), test2.txt(1KB), md5_hash.txt(1KB), fciv.exe(83KB) and Suspicious_File (1274KB).

These files were located in the *C:\..\input path...*.

There were other options that could be used for more efficient matching of hash values, they were:

- ‘-i|I’ by specifying the threshold of the file sizes
- ‘-jnn’ by generating hash values in multithread method instead sequentially
- ‘-o’ by specifying what file type to hash

When using ‘-i’ option, it would be more efficient if all files were a particular size that was less than specified threshold value. This would only generate the hash values for only those files under the threshold. With using ‘-jnn’, this would allow for a large file or many sets of files to be hashed and would generate hash values in a multithread method. It is more efficient because it would create one producer thread to scan the file system and one hashing thread per CPU core² of the machine performance.

² www.md5deep.sourceforge.net/md5deep.html

By using '-o', one could specify what file type to hash instead of generating hash values for all files.

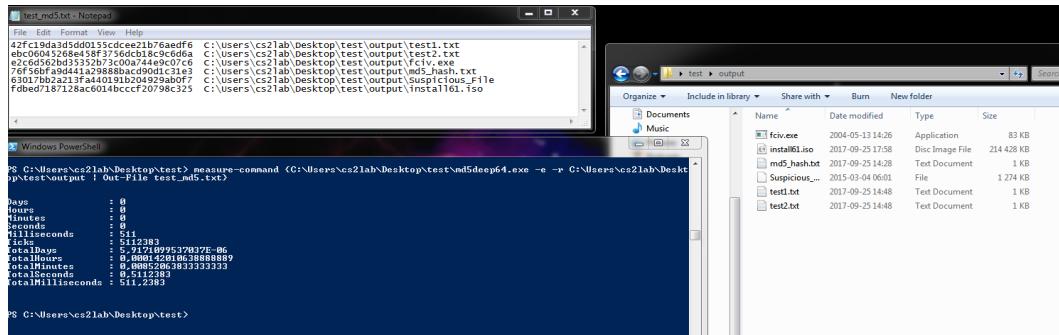
- **Which hashing algorithms did you use? How long did each algorithm take to run?**

We used md5 and sha256 algorithms to hash our test files. The test files used for this timing algorithms test are install61.iso (214428 KB), test1.txt (1KB), test2.txt(1KB), md5_hash.txt(1KB), fciv.exe(83KB) and Suspicious_File (1274KB).

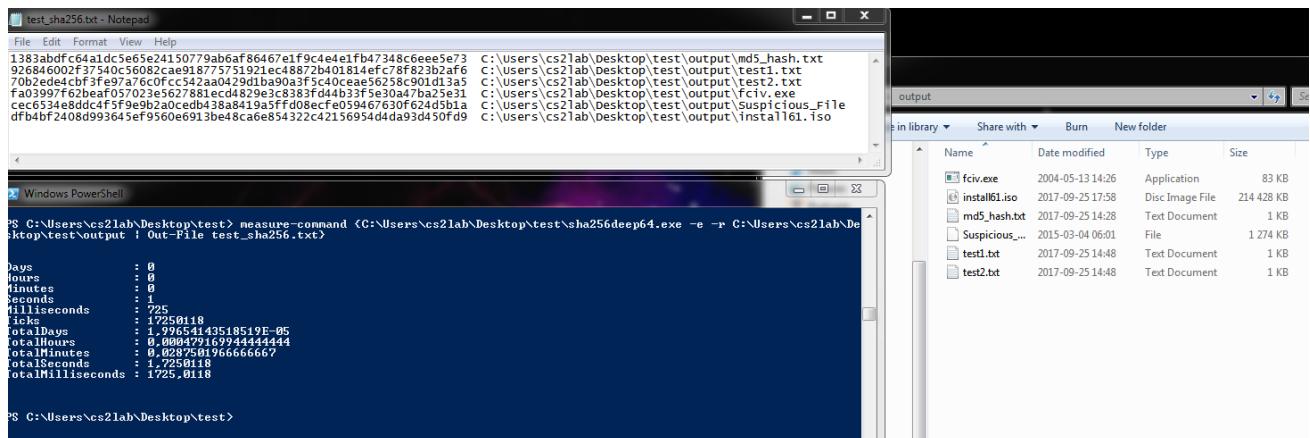
With md5 algorithm took 511milliseconds where sha256 algorithm took 1725milliseconds.

Example commands used were:

```
> measure-command {C:\..\input path...\md5deep64.exe -r C:\..\input folder path...\ | Out-File test_md5.txt}
```



```
> measure-command {C:\..\input path...\sha256deep64.exe -r C:\..\input folder path...\ | Out-File test_sha256.txt}
```



Exercise 2 - File Headers

Objective: Acquire data by acquisition to ensure the integrity of evidence is read-only prior to analyzing the data.

Chain-of-custody: generated hash values for all evidence acquired.

Tools used: We used these TrID version 1.80, AccessData FTK Imager version 3.4.2.6, HexEdit version 4.0 and Exiftool version 10.6.1.0 to identify the filer header and footer information. All twelve files' header hexadecimal values were identified by using the Gary Kessler File Signature Table³.

With File: 01

Result for file 01: We were able to identify this file to be a JPEG file because the file header hexadecimal values (FF D8 FF E0 00 10 4A 46 49 46 00) matched to the Gary Kessler File Signature table. But with TrID, it provided matching based on percentage of accuracy. There were four matching found for file '01' using TrID. 38.1 % and 28.6% of JPG file type matching but there were also 23.8% and 9.5% matching to MP3 file type. See table below.

Tools	File type	Header	Footer/trailer
FTK imager	JFIF - JPEG	FF D8 FF E0 00 10 4A 46 49 46 00	FF D9
HexEdit	JFIF - JPEG	FF D8 FF E0 00 10 4A 46 49 46 00	FF D9
TrID	38.1% matching JPG 28.6% matching JPG 23.8% matching MP3 9.5% matching MP3	N/A	N/A
Exiftool	JPEG	N/A	N/A

FTK Imager with mode set to hex-viewer automatic, a picture of WWII airplanes. To confirm that this file was a jpeg file, we renamed the 01 file to '01.jpg', see figure 1.

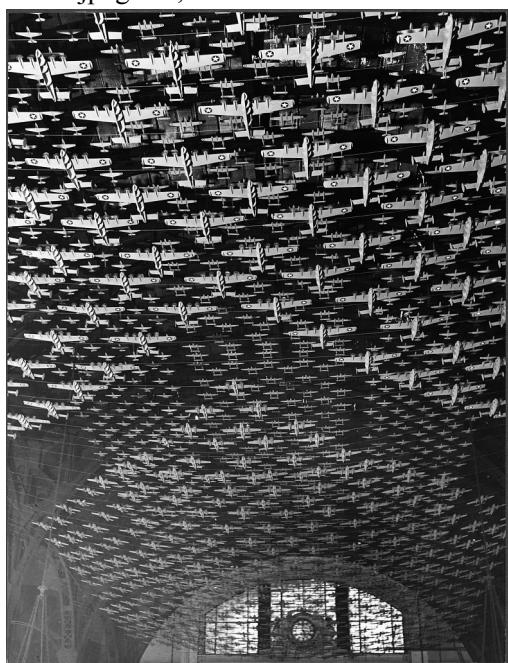


Figure-1: WWII airplanes.

With File: 02

Result for file 02: We were able to identify this file to be a GIF file, see table below. With FTK Imager: when

³ http://www.garykessler.net/library/file_sigs.html

mode was set to hex-view automatic, ‘02’ file was shown as GIF file type with a picture of Bipolar Direct-Current. See table below:

Tools	File type	Header	Footer/trailer
FTK Imager	GIF87ag	47 49 46 38 37 61 67 01	00 3B (:)
HexEdit	GIF87ag	47 49 46 38 37 61 67 01	00 3B (:)
TrID:	60% matching GIF, 30% matching GIF	N/A	N/A
ExifTool	GIF	N/A	N/A

Also to confirm, we renamed the ‘02’ file to ‘02.gif’ and it opened up to:

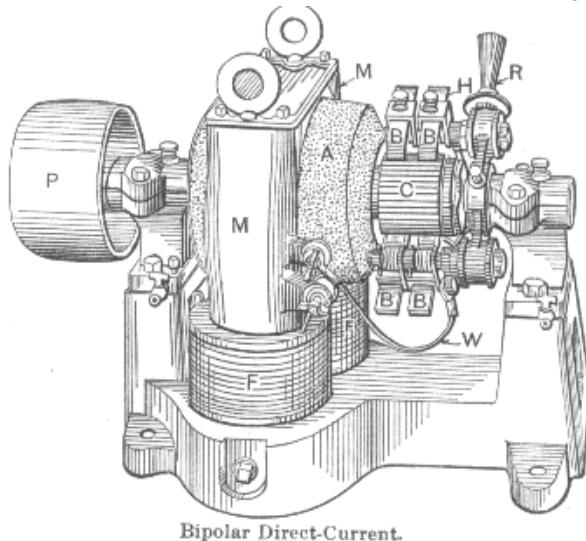


Figure-2: 02.gif

With File: 03

Result for file 03: We were able to identify this file to be an internet shortcut because according to Gary Kessler File Signature Table, there was no file type matching the file header hexadecimal information that we had retrieved from FTK Imager and HexEdit. But according to TrID, it found matching to URL for 91.7% and 8.3 for INI file type. See table below.

Tools	File type	header	Footer/trailer
FTK Imager Note: mode automatic displays as: [InternetShortcut] URL=http://www.dc3.mil/challenge/ Modified=70C990463628CB0139	[InternetShortcut]	5B 49 6E 74 65 72 6E 65-74 53 68 6F72 74 63 75 74 5D	No footer
HexEdit [InternetShortcut] URL=http://www.dc3.mil/challenge/ Modified=70C990463628CB0139	[InternetShortcut]	5B 49 6E 74 65 72 6E 65-74 53 68 6F72 74 63 75 74 5D	No footer
TrID:	91.7% matching URL; 8.3% matching INI	Windows URL shortcut INI Generic Configuration	N/A
ExifTool	Unknown file type err	N/A	N/A

Note: [InternetShortcut] URL=http://www.dc3.mil/challenge/..Modified=70C990463628VB0139File04

With File: 04

Result for file 04: We were able to identify this file to be a PK Zip file. With FTK Imager mode set to hex-viewer automatic, it revealed as PK file type from the file header. See table below.

Tools	File type	header	Footer/trailer
FTK Imager	PK	50 4B 03 04	50 4B
HexEdit	PK	50 4B 03 04	50 4B

			00 00 00 = ...
TrID:	100% matching ZIP	ZIP compressed archive	N/A
Exiftool	ZIP	N/A	N/A

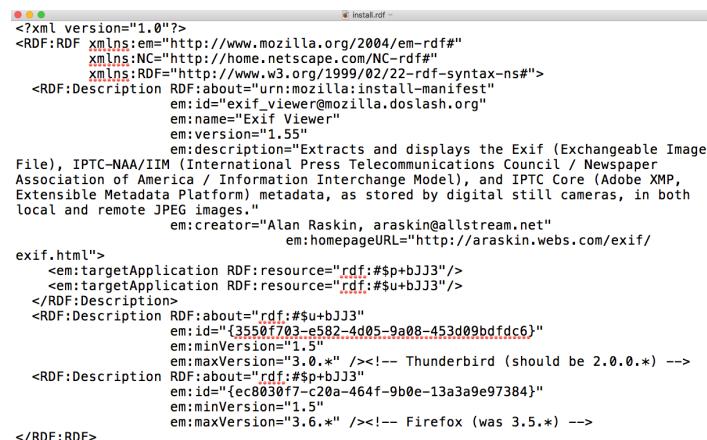
Also, to confirm we renamed '04' file to '04.zip' and opened the zip package. We discovered the 04.zip file contained *exif.exe*, *ReadMe.txt* and *exif.chm*. We speculated the user who used this exif.exe application wanted to obtain metadata⁴ information from a still camera image. The metadata information might contain camera and picture setting including GPS information.

With File: 05

Result for file 05: We were able to identify this file to be an XPI. Any file with this file type was an add-on for any browser. To confirm this, we renamed the 05 file to '05.zip' and extracted the zip which it contained several files and folders. See table below.

Tools	File type	header	Footer/trailer
FTK Imager	PK	50 4B 03 04	00 00 00 = ...
HexEdit	PK	50 4B 03 04	00 00 00 = ...
TrID:	66.6% = XPI, 33% = ZIP, 0.1% = CEL	Mozilla Firefox Browser extension ZIP compression archive Autodesk FLIC Image File extension	N/A
Exiftool	ZIP	N/A	N/A

From this file '*install.rdf*', it explained that this package was an extension for Mozilla Firefox and the name was "Exif Viewer", see figure-3 below:



```
<?xml version="1.0"?>
<RDF:RDF xmlns:em="http://www.mozilla.org/2004/em-rdf#"
           xmlns:NC="http://home.netscape.com/NC-rdf#"
           xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <RDF:Description RDF:about="urn:mozilla:install-manifest"
    em:id="exif_viewer@mozilla.doslash.org"
    em:name="Exif Viewer"
    em:version="1.55"
    em:description="Extracts and displays the Exif (Exchangeable Image
      File), IPTC-NAA/IIM (International Press Telecommunications Council / Newspaper
      Association of America / Information Interchange Model), and IPTC Core (Adobe XMP,
      Extensible Metadata Platform) metadata, as stored by digital still cameras, in both
      local and remote JPEG images."
    em:creator="Alan Raskin, araskin@allstream.net"
    em:homepageURL="http://araskin.webs.com/exif/
  exif.html">
    <em:targetApplication RDF:resource="rdf:#$p+bJJ3"/>
    <em:targetApplication RDF:resource="rdf:#$u+bJJ3"/>
  </RDF:Description>
  <RDF:Description RDF:about="rdf:#$u+b113"
    em:id="{3550ff703-e582-4d05-9a08-453d09bdfdc6}"
    em:minVersion="1.5"
    em:maxVersion="3.0.*" /<!-- Thunderbird (should be 2.0.0.*)-->
  <RDF:Description RDF:about="rdf:#$p+bJJ3"
    em:id="{ec8030f7-c20a-464f-9b0e-13a3a9e97384}"
    em:minVersion="1.5"
    em:maxVersion="3.6.*" /<!-- Firefox (was 3.5.*)-->
</RDF:RDF>
```

Figure-3: "install.rdf" content.

We speculated the user who used this exif.exe application wanted to obtain metadata information from a still camera image. The metadata information might contain camera and picture setting including GPS information.

With File: 06

Result for file 06: We were able to identify this file to be a DMG file for the Mac. See table below:

Tools	File type	header	Footer/trailer
FTK Imager	DMG	At top get this:78 01 63 60 found this at : 3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22 31 2E 30 22 3F 3E <i>Gary Kessler Signature File Table:</i> -78 01 73 0D 62 62 60 is DMG;	No footer

⁴ <http://araskin.webs.com/exif/exif.html>

		-3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22 31 2E 30 22 3F 3E is XML	
HexEdit	DMG	Got:78 01 63 60 <i>Gary Kessler Signature File Table:</i> 3C 3F 78 01 73 0D 62 62 60	No footer
TrID: Marco Pontello marcopon@gmail.com	50% DMG 50% XMI	Disk Image (Macintosh) compressed XML XMill is an efficient XML data compressor	N/A
Exiftool	Unknown file type	N/A	N/A

Tracing through HexEdit on row e92f0, we found xml information pertaining to the Apple ‘XML property list’ which was explained from this link <http://newosxbook.com/DMG.html>. To confirm, we renamed the 06 file to ’06.dmg’ and opened it from a Macintosh. This dmg package contained ‘Hex Fiend.app’.

```

e92d0 BB A3 91 99 81 F5 F4 DB-99 F7 A1 E2 0D 40 75 20 >> 80U-+;à·øu
e92e0 FC E9 71 30 4C 2B 90 37-0A 06 6B 08 00 00 85 98 üéq0L+-7-k-----
e92f0 12 AC 3C 3F 78 6D 6C 20-76 65 72 73 69 6F 6E 3D -<?xml version=
e9300 22 31 2E 30 22 20 65 6E-63 6F 64 69 6E 67 3D 22 "1.0" encoding="UTF-8"?>.<!DOCTY
e9310 55 54 46 2D 38 22 3F 3E-0A 3C 21 44 4F 43 54 59 PE plist PUBLIC
e9320 50 45 20 70 6C 69 73 74-20 50 55 42 4C 49 43 20 "-//Apple//DTD P
e9330 22 2D 2F 2E 41 70 70 6C-65 2F 2F 44 54 44 20 50 LIST 1.0//EN" "h
e9340 4C 49 53 54 20 31 2E 30-2F 2F 45 4E 22 20 22 68 e9350 74 74 70 3A 2F 2F 77 77-77 2E 61 70 70 6C 65 2E http://www.apple.
e9360 63 6F 6D 2F 44 54 44 73-2F 50 72 6F 70 65 72 74 com/DTDs/Propert
e9370 79 4C 69 73 74 2D 31 2E-30 2E 64 74 64 22 3E 0A yList-1.0.dtd">.
e9380 3C 70 6C 69 73 74 20 76-65 72 73 69 6F 6E 3D 22 <plist version="1.0">
e9390 31 2E 30 22 3E 0A 3C 64-69 63 74 3E 0A 09 3C 6B <dict>.<k
e93a0 65 79 3E 72 65 73 6F 75-72 63 65 2D 66 6F 72 6B ey>resource-fork
e93b0 3C 2F 6B 65 79 3E 0A 09-3C 64 69 63 74 3E 0A 09 </key>.<dict>..<key>blkx</key>
e93c0 09 3C 6B 65 79 3E 62 6C-6B 78 3C 2F 6B 65 79 3E <key>blkx</key>

```

Figure 4: 06.dmg

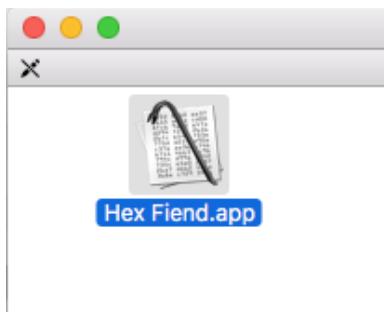


Figure-5: 06.dmg content

This Hex Fiend.app was a hex editor application for a Macintosh.

With File: 07

Result for 07 file: we were able to identify this file to be a RPM file that was a Red Hat Linux Package Manager file. See table below:

Tools	File type	header	Footer/trailer
FTK Imager	HTTP	68 74 74 70	No footer
HexEdit	HTTP	68 74 74 70	No footer
TrID:	100% RPM Package generic	RedHat Package Manager file since seeing several files nested inside from reading the hex	N/A
Exiftool	Unknown file type	N/A	N/A

To confirm this 07 file, we opened 07 file on Kali and confirmed it was RPM file. This file package contained directories of \etc, \usr and \var, (see figure 6 below), where each of these folders contained further nested

directories for Apache server files.

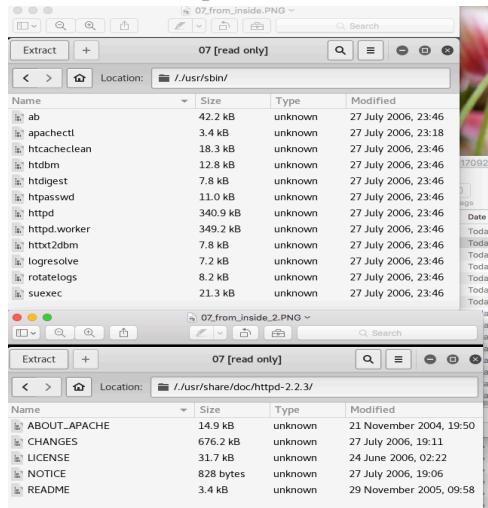


Figure-6: 07.rpm content

With File: 08

Result for 08 file: we were able to identify this file to be a CHM file from Microsoft Compiled HTML. See table below:

Tools	File type	header	Footer/trailer
FTK Imager	ITSF	49 54 53 46 <i>Gary Kessler Signature File Table:</i> Microsoft Compiled HTML Help File	No footer
HexEdit	ITSF	49 54 53 46 <i>Gary Kessler Signature File Table:</i> Microsoft Compiled HTML Help File	No footer
TrID: Davide "Airex" Airaghi airex"AT"tiscali"DOT".it	100% CHM file	Microsoft Compiled HTML Help File	No footer
Exitool Language code: English (U.S.)	CHM	N/A	N/A

To confirm, we downloaded CHM viewer and able to see that '08' file was a chm file for Nvidia Control Panel informaton.

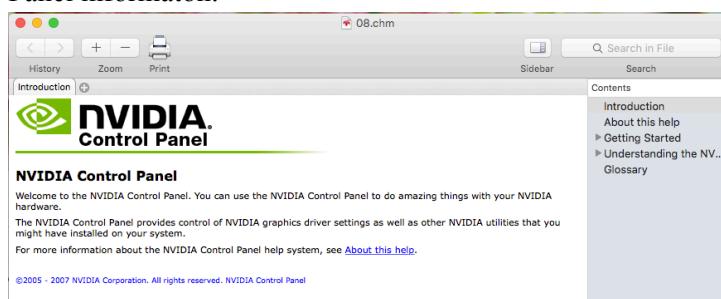


Figure-7: 08.chm content

With File: 09

Result for 09 file: we were able to identify this file to be a MIDI file from FTK Imager, HexEdit and TrID. To confirm, we renamed 09 file to '09.mid' and able to play the music. It was a MIDI sound file.

Tools	File type	header	Footer/trailer
FTK Imager	MThd MIDI	4D 54 68 64 <i>Gary Kessler Signature File Table:</i> Musical Instrument Digital Interface (MIDI) sound file	No footer

HexEdit	MThd MIDI	49 54 53 46 <i>Gary Kessler Signature File Table:</i> Musical Instrument Digital Interface (MIDI) sound file	No footer
TrID:	100% MIDI	MIDI Music URL: www.midi.org	No footer
exiftool	Unknown file type	N/A	N/A

With File: 10

Result for 10 file: we were able to identify this file to be a ReadMe file from FTK Imager and HexEdit. From Gary Kessler Signature File Table, there was no file type for '.txt'. With FTK Imager mode set to hex view automatic, we were able to see a README file of Microsoft File Checksum Integrity Verifier V2.05. See table below:

	File type	header	Footer/trailer
FTK Imager	A readme file from Microsoft File Checksum Integrity	4D 69 63 72 6F 73 6F 66 Microsoft (R) File Checksum Integrity Verifier V2.05 README file <i>Gary Kessler Signature File Table:</i> Microsoft type of file when have these 4D 69 63 72 as the 4 bytes	No footer
HexEdit	A readme file from Microsoft File Checksum Integrity	4D 69 63 72 6F 73 6F 66 Microsoft (R) File Checksum Integrity Verifier V2.05 README file <i>Gary Kessler Signature File Table:</i> Microsoft type of file when have these 4D 69 63 72 as the 4 bytes	No footer
TrID:	0.0% matched	N/A	N/A
exiftool	Unknown file type	N/A	N/A

To confirm, we renamed 10 file to '10.txt' and opened it. It was a ReadMe file for File Checksum Integrity Verifier. But we tried to rename to 10 file to '10.doc' or '10.docx' and was not able to open the file.

With File: 11

Result for 11 file: we were able to identify this file to be a Microsoft Word doc file from TrID and Exiftool. With FTK Imager mode set to hex-viewer to text, we were able to see the information about '*Robocopy is a 32 bit command-line tool*' information. Also when we renamed the 11 file to '11.doc' and it opened up as a windows word doc. It was a user manual of Robocopy.exe (Robust File Copy Utility version XP010). See table below:

Tools	File type	header	Footer/trailer
FTK Imager	OLE	D0 CF 11 E0 A1 B1 1A E1 <i>Gary Kessler Signature File Table:</i> An Object Linking and Embedding (OLE) format known as Compound Binary File format by Microsoft used by Microsoft Office	No footer
HexEdit	OLE	D0 CF 11 E0 A1 B1 1A E1 <i>Gary Kessler Signature File Table:</i> An Object Linking and Embedding (OLE) format known as Compound Binary File format by Microsoft used by Microsoft Office	No footer
TrID:	36% DOC 33.7 % XLS 21.3% DOC 9%	Microsoft Word Doc Microsoft Excel sheet Microsoft Word doc (old ver) Generic OLE2/Multistream Compound File	No footer
Exiftool	DOC	N/A	N/A

With File: 12

Result for 12 file: we were able to identify this file to be a Bit Torrent file from TrID and Exiftool. See table below:

<i>Tools</i>	<i>File type</i>	<i>header</i>	<i>Footer/trailer</i>
FTK Imager	N/A	64 38 3A 61 6E 6E 6F 75 Gary Kessler has no info with this header decimal value	N/A
HexEdit	N/A	64 38 3A 61 6E 6E 6F 75 Gary Kessler has no info with this header decimal value	N/A
TrID:	57.7% TORRENT 42.3% TORRENT	Bit Torrent Link (Trackerless) Bit Torrent Link	N/A
ExifTool	Torrent	Note: file extension: torrent	N/A

From FTK Imager and HexEdit, we found these info:

d8:announce39:http://torrent.ubuntu.com:6969/announce13:announce-list1139:http://torrent.ubuntu.com:6969/announceel44:http://ipv6.torrent.ubuntu.com:6969/announceee7:comment29:Ubuntu CD releases.ubuntu.com13:creation
datei1256817676e4:infod6:lengthi724353024e4:name29:ubuntu-9.10-desktop-amd64.iso12:piece
lengthi524288e6:pieces27640:1
Ubunte CD release from 2009 name29:ubuntu-9.10-desktop-amd64.iso

Exercise 3 – Anti Forensics

Objective: Acquire data by acquisition to ensure the integrity of evidence is read-only prior to analyzing the data.

Evidences: To acquire the c.mp3 and Suspicious_File evidence files by using file carving technique to investigate what these two files really are.

Chain-of-custody: generated hash values for all evidence acquired from tools.

Tools used: EnCase version 8.04, AccessData FTK Imager version 3.4.2.6, GpgEX part of Gpgwinversion 3.0.0, Elcomsoft Forensic Disk Decryptor version 1.12.324.3299, HexEdit version 4.0, Scalpel, Foremost, TrID version 1.80, Exiftool version 10.6.1.0, Malwaretracker.com and Virustotal.com.

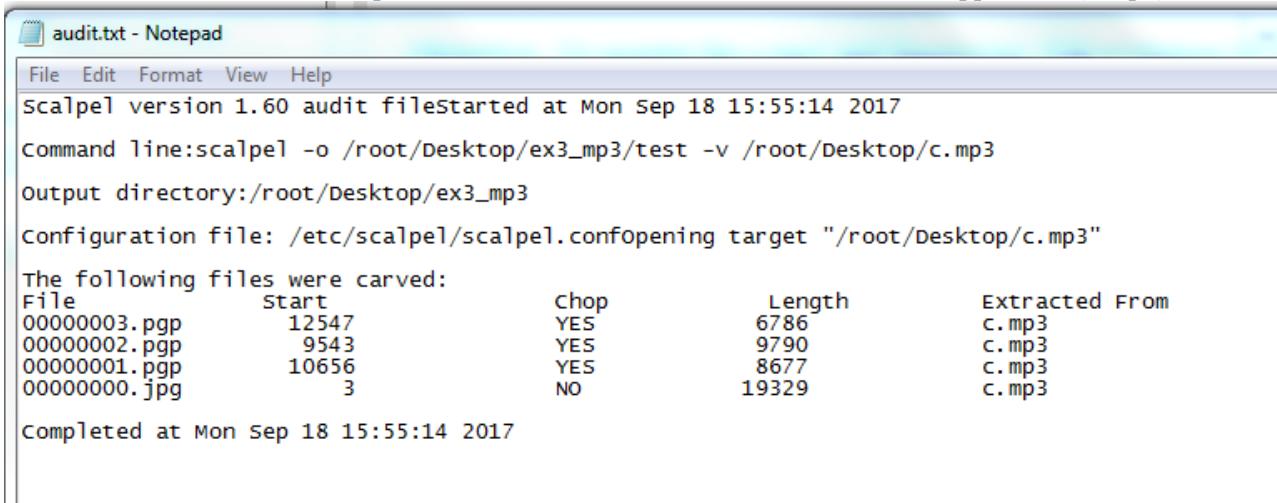
Note: EnCase, Scalpel and Foremost automatically performed the file carving on these two evidence files.

I) Acquisition: c.mp3

How did you proceed your examination and analysis of the files? Describe methodology and your results

From Kali(linux), commands used for c.mp3 file

- scalpel -o /root/Desktop/ex3_mp3/test -v /root/Desktop/c.mp3
-
- Resulted with extraction of c.mp3 file into one JPG and three of PGP files. See appendix 1(c.mp3).



The screenshot shows a Notepad window titled "audit.txt - Notepad". The content of the window is a Scalpel audit log. It starts with the header "Scalpel version 1.60 audit file started at Mon Sep 18 15:55:14 2017", followed by the command line used: "Command line:scalpel -o /root/Desktop/ex3_mp3/test -v /root/Desktop/c.mp3". It then specifies the output directory: "Output directory:/root/Desktop/ex3_mp3" and the configuration file: "Configuration file: /etc/scalpel/scalpel.conf opening target '/root/Desktop/c.mp3'". The next section, "The following files were carved:", lists the extracted files with their details:

File	Start	Chop	Length	Extracted From
00000003.pgp	12547	YES	6786	c.mp3
00000002.pgp	9543	YES	9790	c.mp3
00000001.pgp	10656	YES	8677	c.mp3
00000000.jpg	3	NO	19329	c.mp3

The log concludes with "Completed at Mon Sep 18 15:55:14 2017".

Figure-8: Summary Scalpel report on c.mp3 file.

- The 00000000.jpg file was a picture of Keira Knightley. See figure-9 below.



Figure-9: Result of 00000000.jpg file.

- With three PGP files, we were not able to decrypt using this GpgEX tool.
- PGP files not able to decrypt because GpgEX tool required a key to decrypt a pgp file.
- Foremost tool on Kali:
 - Foremost -o /root/Desktop/ex3_mp3/test -i /root/Desktop/c.mp3
 - Extraction result was only one file. It was a 00000000.jpg file. Opening this extracted 00000000.jpg file revealed it was the same picture as from scalpel extracted 00000000.jpg file.

Analysis:

- Windows: we used HexEdit and TrID to analyze further this c.mp3. It appeared to be that the c.mp3 file header has been modified with '49 44 33' to hide the real file type which was JFIF. But according to TrID, it reported 100% matching to mp3 file.

Tools	File type	header	Footer/trailer
HexEdit	JFIF	49 44 33 FF D8 FF E0 00 10 4A 46 49 46 49 46 00 '4A 46 49 46' = JFIF Gary: FF D8 FF E0 xx xx 4A 46 ÿØÿà..JF 49 46 00 IF. JFIF, JPE, JPEG, JPG JPEG/JFIF graphics file	FF D9
FTK Imager	No data found	N/A	N/A
ExifTool	File Format Error	N/A	N/A
TrID:	100% matching MP3	Note: not a real mp3 file 49 44 33 : only read the 3 bytes from header info and assume it is mp3	N/A

Malwaretracker.com/doc.php was used to analyze further of c.mp3 file:

Md5: 670a8c0db494ced4882e44b27dbd6af2
 Sha1: c5e6e5e1715e90879829264c88ad83160bbe358c
 Sha256:
 Content/type: audio ID3 type verion 2.255.216
 Analysis time: 5.2s
 Result: **clean**

Virustotal.com was used to analyze further of c.mp3 file:

Md5: 670a8c0db494ced4882e44b27dbd6af2
 Sha1: c5e6e5e1715e90879829264c88ad83160bbe358c
 FileType: MP3
 ID3Size 0
 MIMETYPE audio/mpeg
 Warning: **invalid ID3 header**

Summary of c.mp3 file:

Could you identify the files? If so, what type of files were they? Could you find anything peculiar about the files?

It appeared to be an image JPEG file but not a mp3 file. Based on the Virustotal.com warning message, the **invalid ID3 header** confirmed with HexEdit that the first 3 bytes of header information were modified to be a mp3 file to hide the real purpose of this JPEG file. Also, the extension was modified to '.mp3' instead of '.jpg'.

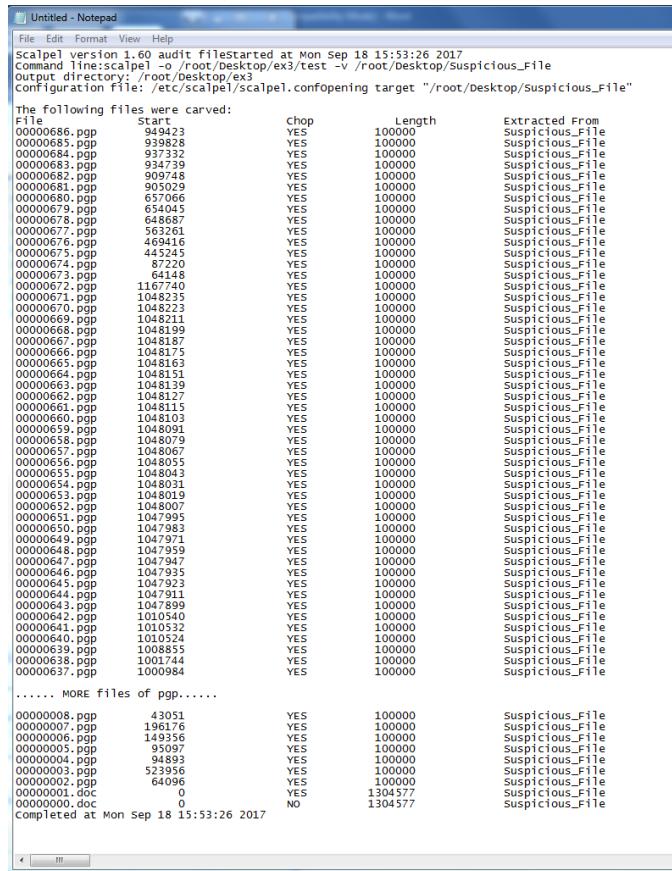
II) Acquisition: Suspicious_File

How did you proceed your examination and analysis of the files? Describe methodology and your results.

- 1) Scalpel from Kali(linux) commands was used for Suspicious_File file:

- scalpel -o /root/Desktop/ex3_mp3/test -v /root/Desktop/ Suspicious_File

Resulted with extraction of Suspicious_File file into 687 files where two files (00000000.doc and 00000001.doc) were doc files and the rest were PGP files. See appendix 2 (Suspicious_File).



```

Untitled - Notepad
File Edit Format View Help
Scalpel version 1.60 audit file started at Mon Sep 18 15:53:26 2017
Command line:scalpel -o /root/Desktop/ex3/test -v /root/Desktop/Suspicious_File
Output directory: /root/Desktop/ex3
Configuration file: /etc/scalpel/scalpel.conf opening target "/root/Desktop/Suspicious_File"
The following files were carved:
File Start Chop Length Extracted From
000000686.pgp 93423 YES 100000 Suspicious_File
000000685.pgp 939828 YES 100000 Suspicious_File
000000684.pgp 937332 YES 100000 Suspicious_File
000000683.pgp 934739 YES 100000 Suspicious_File
000000682.pgp 934738 YES 100000 Suspicious_File
000000681.pgp 905029 YES 100000 Suspicious_File
000000680.pgp 657066 YES 100000 Suspicious_File
000000679.pgp 654045 YES 100000 Suspicious_File
000000678.pgp 648687 YES 100000 Suspicious_File
000000677.pgp 597201 YES 100000 Suspicious_File
000000676.pgp 469416 YES 100000 Suspicious_File
000000675.pgp 445245 YES 100000 Suspicious_File
000000674.pgp 87220 YES 100000 Suspicious_File
000000673.pgp 64148 YES 100000 Suspicious_File
000000672.pgp 110470 YES 100000 Suspicious_File
000000671.pgp 1048235 YES 100000 Suspicious_File
000000670.pgp 1048232 YES 100000 Suspicious_File
000000669.pgp 1048211 YES 100000 Suspicious_File
000000668.pgp 1048199 YES 100000 Suspicious_File
000000667.pgp 1048187 YES 100000 Suspicious_File
000000666.pgp 1048175 YES 100000 Suspicious_File
000000665.pgp 1048163 YES 100000 Suspicious_File
000000664.pgp 1048151 YES 100000 Suspicious_File
000000663.pgp 1048139 YES 100000 Suspicious_File
000000662.pgp 1048127 YES 100000 Suspicious_File
000000661.pgp 1048115 YES 100000 Suspicious_File
000000660.pgp 1048103 YES 100000 Suspicious_File
000000659.pgp 1048091 YES 100000 Suspicious_File
000000658.pgp 1048079 YES 100000 Suspicious_File
000000657.pgp 1048067 YES 100000 Suspicious_File
000000656.pgp 1048055 YES 100000 Suspicious_File
000000655.pgp 1048043 YES 100000 Suspicious_File
000000654.pgp 1048031 YES 100000 Suspicious_File
000000653.pgp 1048019 YES 100000 Suspicious_File
000000652.pgp 1048007 YES 100000 Suspicious_File
000000651.pgp 1047995 YES 100000 Suspicious_File
000000650.pgp 1047983 YES 100000 Suspicious_File
000000649.pgp 1047971 YES 100000 Suspicious_File
000000648.pgp 1047959 YES 100000 Suspicious_File
000000647.pgp 1047947 YES 100000 Suspicious_File
000000646.pgp 1047935 YES 100000 Suspicious_File
000000645.pgp 1047933 YES 100000 Suspicious_File
000000644.pgp 1047911 YES 100000 Suspicious_File
000000643.pgp 1047899 YES 100000 Suspicious_File
000000642.pgp 1010540 YES 100000 Suspicious_File
000000641.pgp 1010532 YES 100000 Suspicious_File
000000640.pgp 1010524 YES 100000 Suspicious_File
000000639.pgp 1008855 YES 100000 Suspicious_File
000000638.pgp 1001744 YES 100000 Suspicious_File
000000637.pgp 1000984 YES 100000 Suspicious_File
..... MORE files of pgp.....
000000008.pgp 43051 YES 100000 Suspicious_File
000000007.pgp 196176 YES 100000 Suspicious_File
000000006.pgp 19616 YES 100000 Suspicious_File
000000005.pgp 95097 YES 100000 Suspicious_File
000000004.pgp 94893 YES 100000 Suspicious_File
000000003.pgp 523956 YES 100000 Suspicious_File
000000002.pgp 64096 YES 100000 Suspicious_File
000000001.doc 0 YES 1304577 Suspicious_File
000000000.olec 0 NO 1304577 Suspicious_File
Completed at Mon Sep 18 15:53:26 2017

```

Figure-10: Scalpel results.

- With PGP files, we downloaded Elcomsoft Forensic Disk Decryptor to try to decrypt PGP files. But this tool also asked for the keys to these PGP files in order to decrypt them. We could not process further.
- Foremost tool on Kali:
 - > foremost -i /root/Desktop/ex3/Suspicious_file -o /root/Desktop/test
 - Foremost extracted into one OLE file (00000000.ole).
- EnCase on Windows 7:

Using EnCase, we were able to file carve Suspicious_File and discovered there were two files wrapped inside the Suspicious_File. These two files were 'Details' and 'File_0'. See figure-11 below:

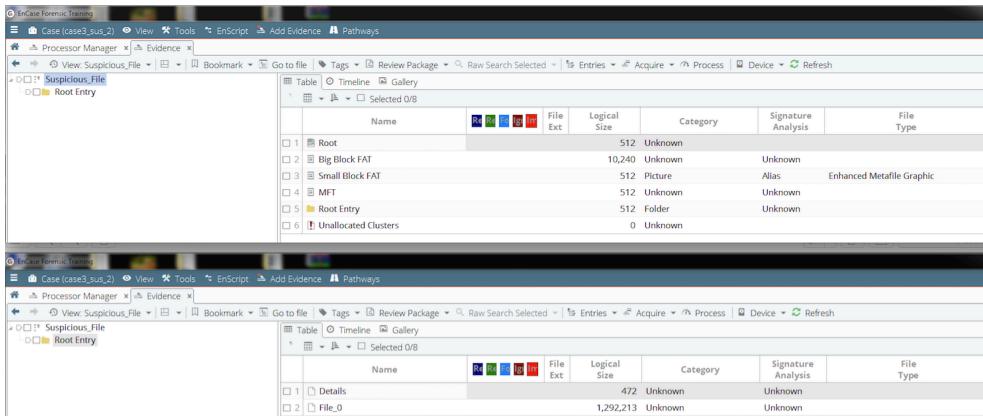


Figure-11: EnCase result.

Analysis: Suspicious_File

Tools	File type	header	Footer/trailer
FTK	OLECF	D0 CF 11 E0 A1 B1 1A E1 <i>Gary Kessler Signature File Table:</i> According to Gary Kessler, it was an OLECF file.	N/A
HexEdit	OLECF	D0 CF 11 E0 A1 B1 1A E1 <i>Gary Kessler Signature File Table:</i> According to Gary Kessler, it was an OLECF file.	N/A
TrID:	100% OLE2	Generic OLE2 Multistream Compound File	N/A
Exiftool	N/A	N/A	N/A

Malwaretracker.com/doc.php site redirected us to this Cryptam.com for analysis of Suspicious_File, see figure-12 below:

Sample Details

```
original filename: Suspicious_File
size: 1304576 bytes
submitted: 2014-01-27 07:39:41
md5: 63017bb2a213fa440191b204929ab0f
sha1: f454953c6ae4496b21d2e4c1006842aff60b90eb
sha256: cec6534e8ddc4f519e9b2a0cedb438a8419a5ffd08ecfe059467630f624d5b1a
ssdeep: 24576:AtgRvu+NB53rj1HXQ+5qj8ie+0QMv4RhDHd91S1etvw4qEY9c4jC1CV+E4cY:ATgRvu+fNB53rj1HA+5qj8iePd91YeX
content/type: Composite Document File V2 Document, No summary info
analysis time: 53.57 s
result: malware [150]
embedded executable: found
signature hits:
2638: string.This program cannot be run in DOS mode
1244088: string.LoadLibraryA
1243996: string.GetModuleHandleA
1244048: string.GetCommandLineA
1246706: string.GetSystemMetrics
1243770: string.GetProcAddress
1243712: string.EnterCriticalSection
1243400: string.CloseHandle
1244398: string.CreateFileA
1247318: string.RegOpenKeyExA
1081060: string.user32.dll
1163676: string.KERNEL32
1161103: string.ExitProcess
1246846: string.GetMessageA
1246552: string.CreateWindowA
dropped.file exe 91babceabb2263975b0c4309e82ca977 / 1302016 bytes / @ 2560
```

Figure-12: Malwaretracker results.



Strings

[raw strings](#)
[decrypted raw strings](#)

Dropped Files

exe at 2560
md5: [91babceabb2263975b0c4309e82ca977](#)
sha1: [bc7bact35c45d6c5b28b1aa319e74bd543bdc82a](#)
sha256: [34b0111f7e5faa2ba83dcab079d82f74b7876e585a1a751a4af92a7b1f4def29](#)
[view strings](#)

Figure-13: Key generated from Cryptam.com to decrypt the hex.

According to Malwaretracker.com analysis of the Suspicious_File, it identified 150 malwares, see figure-12. It also had identified some strings that looked like function calls from a program and identified a '*droppedfile.exe*' program. We also noticed this '*droppedfile.exe*' had its own hash value from Malwaretracker. While analyzing the output from Cryptanalysis section (see figure-13), a key value was found '6a'. We speculated that this key value was used to decrypt the raw string data presented by Malwaretracker of '*droppedfile.exe*'. When accessing this link '[decrypted raw strings](#)' option, we noticed these peculiarities:

- It revealed that there were several files with the word 'copyright' such as:
 - *Copyright (C) 1998, Thomas G. Lane*
 - *deflate 1.1.3 Copyright 1995-1998 Jean-loup Gailly*
 - *inflate 1.1.3 Copyright 1995-1998 Mark Adler*
 - *Copyright (c) 1992-2001 by P.J. Plauger, licensed by Dinkumware, Ltd. ALL RIGHTS RESERVED.*
 - *Copyright 2003 - 2004 PopCap Games, Inc.*
- Additionally, we found these URL of:
 - <http://www.popcap.com/win32updatecheck2.php?prod=>
 - <http://www.popcap.com/register.php?theGame=>
 - <http://www.microsoft.com/directx>
 - http://www.popcap.com/beta_validate.php?prod=
 - <http://store.gamehouse.com/buyzuma.jsp?AID=%AFFIL%&D=%DAYS%&M=%MINUTES%>
- And these information:
 - *OriginalFilename*
 - *Zuma.exe*
 - *ProductName*
 - *Zuma Deluxe*
 - *ProductVersion*
 - *1.0.0.1*
 - *SpecialBuild*
 - *Presented by GameHouse*
- We also discovered there were random information such as:
 - *TimeZoneName=Arab Standard Time*
 - *Names of months but May, June and July months were missing*
 - *Names of days*
 - *Instructions*

- names of players and characters
- names of sounds files
- names of image files and many more.....

Virustotal.com:

According to Virustotal.com, we were able to obtain similar information as EnCase because both tools identified two files, Details and File_0, see figure-14 below.

Virustotal.com identified these information from the *Detection info* tab:

Generic OLE2 Multistream Compound File
Detection: CAT-QuickHeal: OLE.Win32.Agent.EB
NANO-Antivirus: Virus.Win32.Gen.ccmw

From *Details* tab:

Basic Properties	
MDS	63017bb2a213fa440191b204929ab0f7
SHA-1	f454953c64e4496b21d2e4c1006842aff60b90eb
File Type	MS Word Document
Magic	CDF V2 Document, corrupt: Cannot read summary info
SSDeep	24576:ATgRvu+fNBS5r3j1HXQ+5qBlie+IOQMv4RhDhd9151etwww4qEyY9c4jC1CV+E4cY:ATgRvu+fNBS5r3j1HA+5qBliePd91YeX
TRID	Generic OLE2 / Multistream Compound File (100%)
File Size	1.24 MB

Tags	
doc	

History	
First Submission	2011-02-25 12:16:33
Last Submission	2017-09-17 13:20:11
Last Analysis	2017-09-17 13:20:11

File Names	
Suspicious_File	
File	
63017bb2a213fa440191b204929ab0f7	
Suspicious_File2	

OLE Compound File Info	
OLE Streams	
Root Entry	Name: Root Entry Size: 512 Type Literal: root
Details	Name: Details SId: 1 Size: 472 Type Literal: stream
File_0	Name: File_0 SId: 2 Size: 1292213 Type Literal: stream

ExitTool File Metadata	
FileType	FPX
FileTypeExtension	fpx
MIMEType	image/vnd.fpx

Figure-14: Virustotal.com results

We speculated what VirusTotal had detected two types of viruses are: '*OLE.Win32.Agent.EB*' and '*Virus.Win32.Gen.ccmw*'. Additionally, it had identified two files that were similarly named as EnCase results and they were '*Details*' and '*File_0*', see figure-14.

Summary of Suspicious_File:

Could you identify the files? If so, what type of files were they? Could you find anything peculiar about

the files?

We were not really able to identify Suspicious_File because it appeared to contain multiple files that were wrapped into one large file. We were able to carve the files by using Scalpel, Foremost and EnCase. Only Scalpel tool able to carve out 687 files (see appendix-2) where Foremost carved only one file out and EnCase carved two files out. Also we speculated that there could be two viruses stored in the Suspicious_File based on the findings from Virustotal.com and Malwaretracker.com. See Analysis section above regarding the peculiarities found.

Exercise 4 - Acquisition

See *DIFO2017_Lab1_Report_Group14_Ex4.pdf* for answer to Exercise 4.

Exercise 5 - Cracking

Objective: To crack the protected and or encrypted files by using brute force technique.

Evidences: Unnamed 1.xls, casssh.pdf, Untitled.docx, hr.gpg.tar and wallet.dat

Chain-of-custody: Used md5 and sha256 algorithm to hash the evidence files.

Tools used: hashcat version 3.6.0, AccessData PRTK version 8.1.0 build 946 and sha256deep64.exe

Acquisition: The steps to acquire the evidence files used are:

a) Md5deep64.exe and sha256deep64.exe to generate md5 hash values:

Name	MD5	SHA256
casssh.pdf	f77f0c2ea19035ace1d47b266245984d	a5ba8562717a9c128bdb87d3b0b327cea7bb24f33fd599d5de8fc6d41c174d02
ht.zip.tar.gpg	159ef8f471caef32db49dc9b331e10a5	2fe1cd6a80f6609efbc4fc142ce552ef46155f0fe1b0b765f0b12ea7fd5d8f8b
untitled.docx	3a9b988f1496b2598cf08023603bcf3c	c28739a4507ca3f39ac0cf1a06de6f58d410b3a1fb8724268264d1cbe67e8112
wallet1.dat	fbf3b26fc8e9fe4ec2e70175255e36c	6c073f77c40ca84a1ada73452633b58359d055e37ae75dcec0f663538db859a5
Untitled 1.ods	0a4532f87c41c31f1b716cd21ea8fa51	126b46eb0c3891f86f38af95b810ed083f242e5aaaa5bbd84caf9f1ee28af6a3

b) To ensure that we didn't alter these five evidence files, we used the copied of the evidence files to crack the passwords on these files.

1) Casssh.pdf: We used dictionary brute force attack with this evidence file using 'Access Data Password Recovery Toolkit (PRTK)' tool.

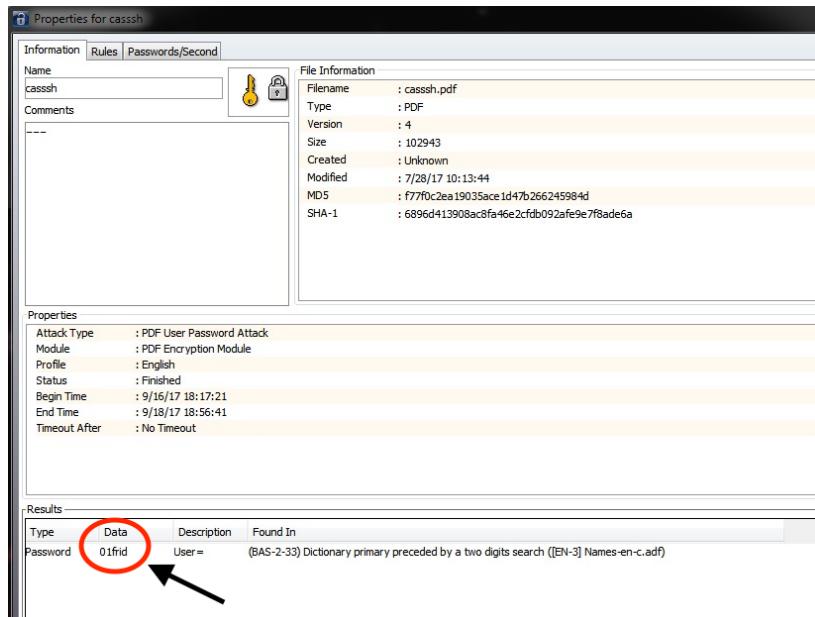


Figure-15: Password for pdf file

It tooks about 8 hours to crack this casssh.pdf and two days to verify the password. The password was '01frid' for this 'casssh.pdf' file. By using this password, we were able to open 'casssh.pdf' file and the result was:



Figure-16: Result of casssh.pdf.

2) ht.zip.tar.gpg: We started the password cracking process on 2017-09-18 14:38:10 using ‘Access Data Password Recovery Toolkit (PRTK)’ and

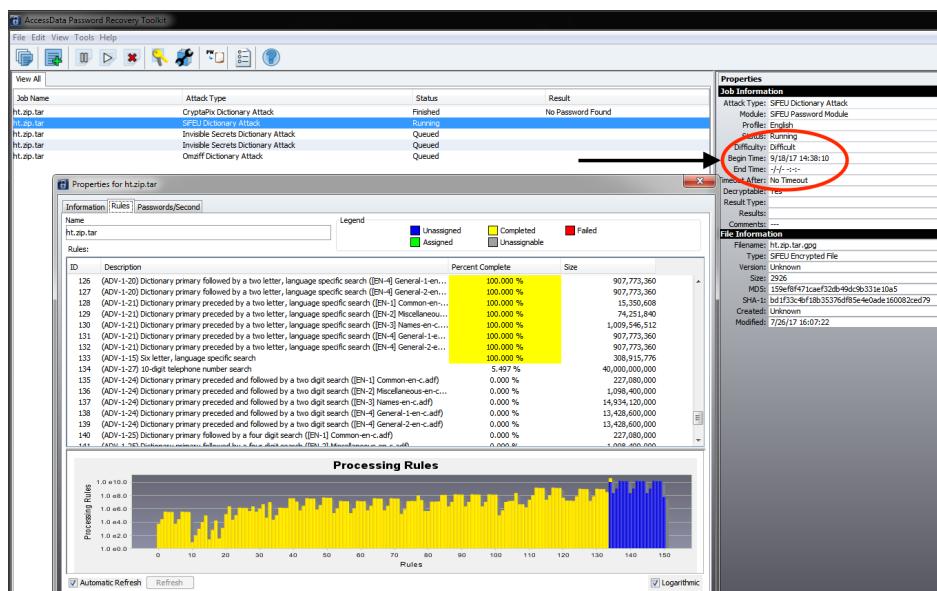
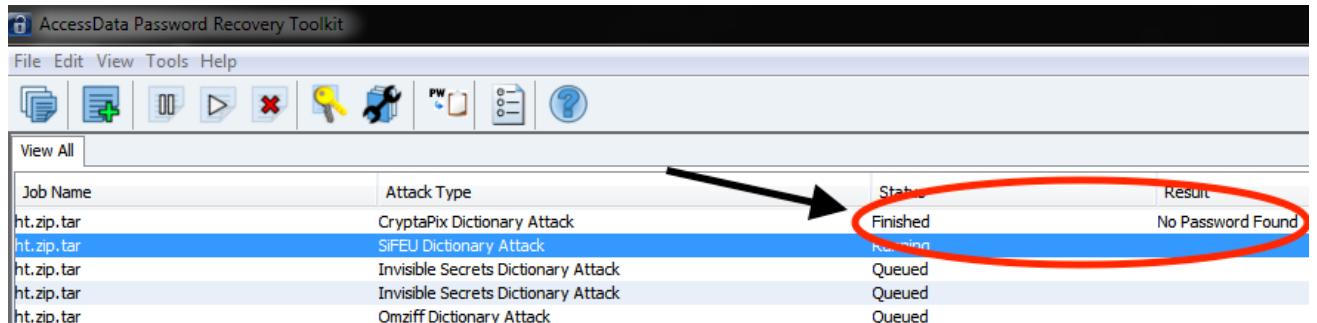


Figure-17: Result of “ht.zip.tar.gpg”

we were not able to complete this file due to taking too long to crack. The reasons were: it had multiple Attack type rule policies that were needed to be processed to crack the password on this file, see figure-17 above.

We could crack with one of the Attack Type rule policy and the result was ‘no password found’ (see figure 18 below) and other Attack type rule policies continued to process which had taken more than two weeks to process.

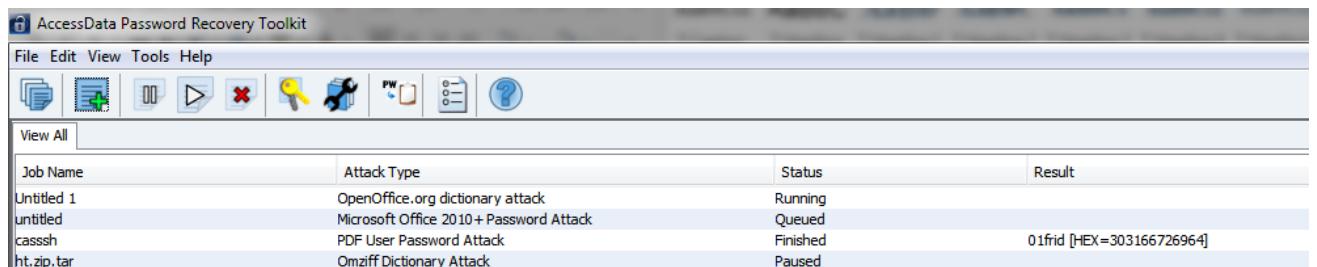


Job Name	Attack Type	Status	Result
ht.zip.tar	CryptaPix Dictionary Attack	Finished	No Password Found
ht.zip.tar	SIEU Dictionary Attack	Running	
ht.zip.tar	Invisible Secrets Dictionary Attack	Queued	
ht.zip.tar	Invisible Secrets Dictionary Attack	Queued	
ht.zip.tar	Omziff Dictionary Attack	Queued	

Figure-18: Finished with one file rule policy

3) Untitled 1.ods, untitled.docx:

We used ‘Access Data Password Recovery Toolkit (PRTK)’ to crack ‘Untitled 1.ods’ and ‘untitled.docx’ files. These files also have been running more than two weeks similar to file ‘ht.zip.tar.gpg’ procedure above and still it has been continuing to process. We were not able to complete these files for password cracking due to limited time resource.



Job Name	Attack Type	Status	Result
Untitled 1	OpenOffice.org dictionary attack	Running	
untitled	Microsoft Office 2010+ Password Attack	Queued	
casssh	PDF User Password Attack	Finished	
ht.zip.tar	Omziff Dictionary Attack	Paused	01frid [HEX=303166726964]

Figure 19: Result of **Untitled 1.ods, untitled.docx**:

- 4) **wallet1.dat:** We used ‘Access Data Password Recovery Toolkit (PRTK)’ to crack ‘wallet1.dat’ and an error message had occurred. The error was “Unable to identify file type for decryption”, see figure-20.

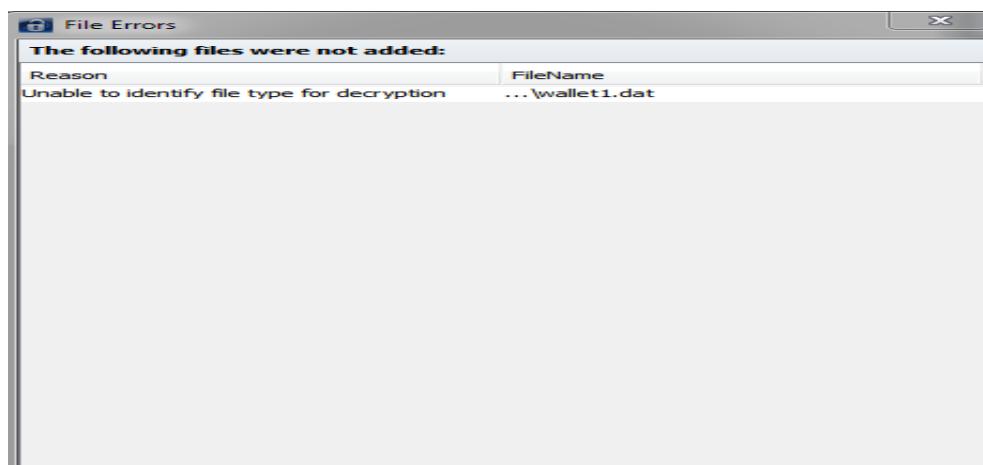


Figure 20: Result of **wallet1.dat**

5) We used PWD2 (10.11.200.152) machine as well to crack this file '*casssh.pdf*' by using Hashcat tool. Command used were:

```
>C:\Users\group 14\Desktop\hashcat-3.6.0>hashcat64 -m 10500 -a 3 "C:\Users\group 14\Desktop\Exercise5_Cracking\casssh_hash"
```

Also, we tried to crack other evidence files such as '*ht.zip.tar.gpg*' using Hashcat on the PWD2 server and it took too long because we were limited with time resource on this server. The other evidence files were not executed using Hashcat.

Exercise 6 - Steganography

Objective: To find any hidden data from the two evidence (*c1l.png* and *c2l.png*) files given.

Evidences: *c1l.png* and *c2l.png*

Chain-of-custody: generated hash values using md5deep64.exe.

Tools used: Exiftool version 10.6.1.0, md5deep64 version 4.4, TweakPNG version 1.4.6, Rizzy.py and Stepic-0.3.

Identify the two files, what they do depict?

From visual inspection, these two files (*c1l.png* and *c2l.png*) depicted CS2 in large black block letters and with smaller blue text of Cyber Systems Security Laboratory on the right side of CS2. See figure 21.

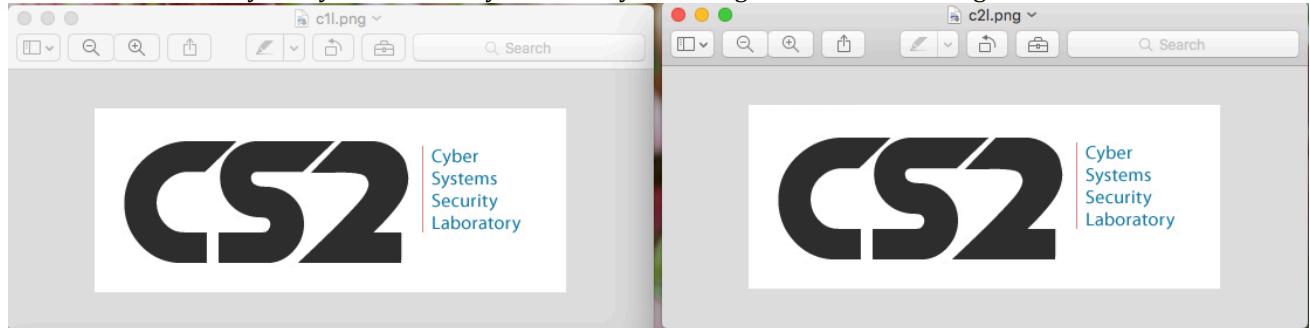


Figure 21: Visual inspection of evidence files.

From the visual inspection, there seem to be no differences between these two files so far.

What are their hash sums?

The hash algorithm used was md5 on both *c1l.png* and *c2l.png* files.

- *c1l.png* hash value: 601450fd443v42f4ece0e3f001ed73b3
- *c2l.png* hash value: b914cc2043a6d5b31ff4bb6f5f1291fc

After we generated the hash values for these two files, we realized there could be differences between these two files since the hash values provided by md5deep64.exe were different.

What are their META data?

Using Exiftool, it identified these two files as unique files because we noticed that *c2l.png* didn't have information of *Significant Bits, Pixels Per Unit X, Pixels Per Unit Y, and Pixel Units* such as the *c1l.png* file. See figure 22.

ExifTool Version Number	
File Name	: 10.61
Directory	: C:/Users/cs2lab/Desktop/test1
File Size	: 18 kB
File Modification Date/Time	: 2017:08:09 14:36:22+02:00
File Access Date/Time	: 2017:09:19 15:47:59+02:00
File Creation Date/Time	: 2017:09:19 15:47:59+02:00
File Permissions	: rw-rw-rw-
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
Pixels Per Unit X	: 7500
Pixels Per Unit Y	: 7500
Pixel Units	: meters
Image Size	: 403x157
Megapixels	: 0.063
-- press RETURN --	
ExifTool Version Number	
File Name	: 10.61
Directory	: C:/Users/cs2lab/Desktop/test1
File Size	: 18 kB
File Modification Date/Time	: 2017:08:09 14:40:16+02:00
File Access Date/Time	: 2017:09:19 15:47:59+02:00
File Creation Date/Time	: 2017:09:19 15:47:59+02:00
File Permissions	: rw-rw-rw-
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
-- press RETURN --	

Figure 22: Metadata information of evidence files.

Which tool/tools did you use to find out if there was any message in the image(s)?

Tools used were: TweakPNG, Exiftool, md5deep64.exe, rizzy.py and stepic.py.

With TweakPNG.exe, it provided the critical chunks information of the evidence files such as IHDR, sBIT, pHYs, IDAT and IEND. See figure-23 which led us to believe that there could be hidden message in one of the evidence file.

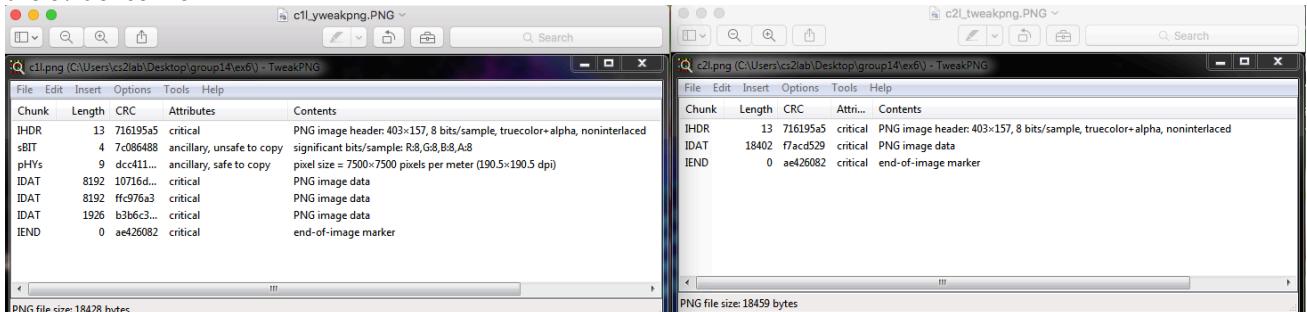


Figure 23: Critical chunks information of evidence files.

When using the Rizzy.py and stepic.py tools, both tools revealed a hidden message from *c2l.png* file which is '<http://xdsas5xcrrxxolc.onion/>'. See figure 24:

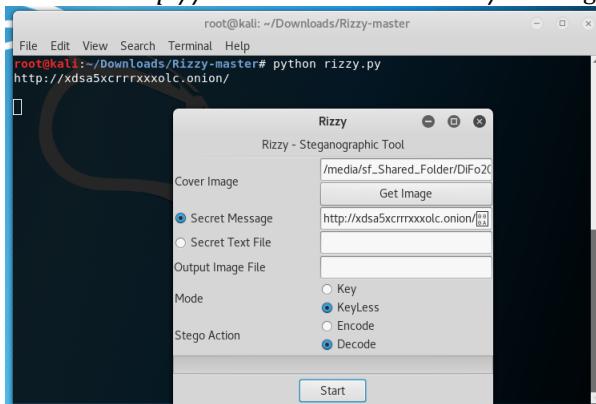


Figure 24: Rizzy.py found hidden message from *c2l.png*.

Additionally, both rizzy.py and stepic.py tools found this '*invalid UTF-8 string passed*' from *c1l.png*. See figure -25:

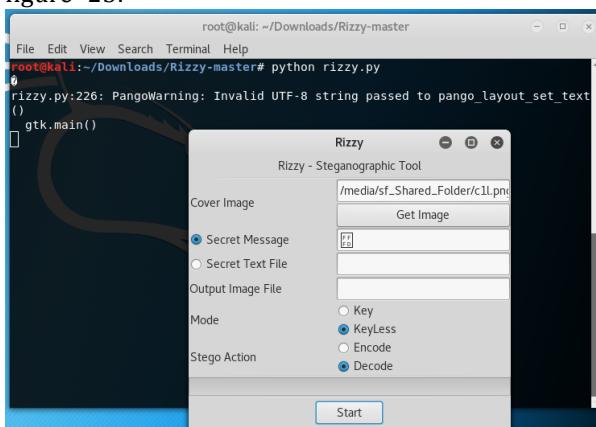


Figure 25: Rizzy.py found invalid UTF-8 message from *c1l.png*.

Analysis:

How was the message hidden in the image?

We speculated that the hidden message was hidden inside the IDAT chunk in the *c2l.png* file. The reason how we derived to this conclusion was that we used rizzy.py to encrypt the hidden message of

'<http://xds5xcrrxxolc.onion/>' from *c2l.png* to *c1l.png*. The test output file *c1l_encoded.png* also had the same critical chunks information like *c2l.png*. See figure-26 and compare to Figure-23.

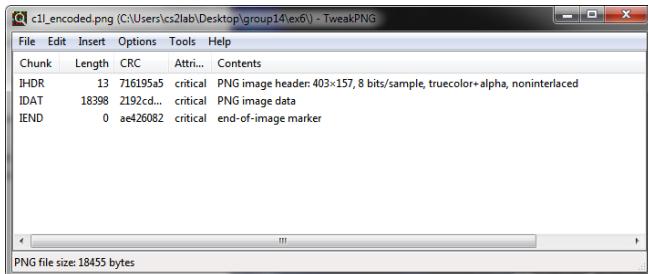


Figure 26: Critical chunks information of *c1l_encoded.png*.

One could see that both *c2l.png* and *c1l_encoded.png* files had the same critical chunks information, see Table-1 below. This confirmed our theory that *c2l.png* had been altered to contain a hidden message and that *c1l.png* was the source for *c2l.png* file. Due to this confirmation test, it allowed us to be able to perform a comparison test.

Table-1: PNG critical chunks information

Critical Chunks	<i>c1l.png</i>	<i>c2l.png</i>	<i>c1l_encoded.png</i> (a test file)
IHDR	Yes	No	No
IDAT	Found 3 chunks	Only 1 IDAT found	Only 1 IDAT
sBit	Yes	No	No
pHYs	Yes	No	No
IEND	Yes	Yes	Yes
Hidden message	Invalid of UTF-8 string	http://xds5xcrrxxolc.onion/	http://xds5xcrrxxolc.onion/

By looking at these critical chunks information, it led us to experiment with another png file (*jaguar.png*) which we downloaded from the internet. We wanted to confirm what a regular png file would have for the critical chunks information so that we could confidently conclude if the *c1l.png* file was an original file and it hadn't been altered in anyway.

By using the *jaguar.png*, we discovered that it did contain the same critical chunks information like *c1l.png* file except that the *jaguar.png* had more IDAT chunks. The extra IDAT chunks might meant that the *jaguar.png* contained more image data than *c1l.png* file. When we encoded the same hidden message from *c2l.png* into *jaguar.png* file, it also revealed that the critical chunks information were same as *c2l.png* file, see Table-2 below.

Table-2: PNG critical chunk information

Critical Chunks	<i>c1l.png</i>	<i>c2l.png</i>	<i>c1l_encoded.png</i> (a test file)	<i>jaguar.png</i>	<i>jaguar_encoded.png</i> (a test file)
IHDR	Yes	No	No	Yes	No
IDAT	Found 3 chunks	Only 1 IDAT found	Only 1 IDAT	38 IDAT found	9 IDAT found
sBit	Yes	No	No	Yes	No
pHYs	Yes	No	No	Yes	No
IEND	Yes	Yes	Yes	Yes	No
Hidden message	Invalid of UTF-8 string	http://xds5xcrrxxolc.onion/	http://xds5xcrrxxolc.onion/	Invalid of UTF-8 string	http://xds5xcrrxxolc.onion/

Could you have found the message in any other way?

Maybe HexExit tool can be used as an alternative tool to find hidden messages in *c2l.png* file.

Appendix 1: c.mp3

```
root@kali:~/Desktop/test# scalpel -o /root/Desktop/test/output_mp3/ -v /root/Desktop/test/c.mp3
```

Scalpel version 1.60

Written by Golden G. Richard III, based on Foremost 0.69.

Output directory: "/root/Desktop/test/output_mp3"

Configuration file: "/etc/scalpel/scalpel.conf"

Coverage maps directory: "/root/Desktop/test/output_mp3"

Opening target "/root/Desktop/test/c.mp3"

Total file size is 19332 bytes

Image file pass 1/2.

Read 19332 bytes from image file.

```
/root/Desktop/test/c.mp3: 100.0% |*****| 18.9 KB 00:00 ETAA jpg header was found at : 3
```

Memory reallocation performed, total header storage = 101

A jpg footer was found at : 19330

Memory reallocation performed, total footer storage = 101

A pgp header was found at : 10656

Memory reallocation performed, total header storage = 101

A pgp header was found at : 9543

Memory reallocation performed, total header storage = 101

A pgp header was found at : 12547

Allocating work queues...

Work queues allocation complete. Building carve lists...

Carve lists built. Workload:

art with header "\x4a\x47\x04\x0e" and footer "\xcf\xc7\xcb" --> 0 files

art with header "\x4a\x47\x03\x0e" and footer "\xd0\xcb\x00\x00" --> 0 files

gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" --> 0 files

gif with header "\x47\x49\x46\x38\x39\x61" and footer "\x00\x3b" --> 0 files

jpg with header "\xff\xd8\xff\xe0\x00\x10" and footer "\xff\xd9" --> 1 files

png with header "\x50\x4e\x47\x3f" and footer "\xff\xfc\xfd\xfe" --> 0 files

bmp with header "\x42\x4d\x3f\x3f\x00\x00\x00" and footer "" --> 0 files

tif with header "\x49\x49\x2a\x00" and footer "" --> 0 files

tif with header "\x4d\x4d\x00\x2a" and footer "" --> 0 files

avi with header "\x52\x49\x46\x46\x3f\x3f\x3f\x41\x56\x49" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x6d\x6f\x6f\x76" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x6d\x64\x61\x74" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x77\x69\x64\x65\x76" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x73\x6b\x69\x70" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x66\x72\x65\x65" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x69\x64\x73\x63" and footer "" --> 0 files

mov with header "\x3f\x3f\x3f\x3f\x70\x63\x6b\x67" and footer "" --> 0 files

mpg with header "\x00\x00\x01\xba" and footer "\x00\x00\x01\xb9" --> 0 files

mpg with header "\x00\x00\x01\xb3" and footer "\x00\x00\x01\xb7" --> 0 files

fws with header "\x46\x57\x53" and footer "" --> 0 files

doc with header "\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00" and footer

"\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00" --> 0 files

doc with header "\xd0\xcf\x11\xe0\xa1\xb1" and footer "" --> 0 files

pst with header "\x21\x42\x4e\x4a\x5\x6f\xb\x5\x4a\x6" and footer "" --> 0 files

dbx with header "\xcf\xad\x12\xfe\xc5\xfd\x74\x6f" and footer "" --> 0 files

idx with header "\x4a\x4d\x46\x39" and footer "" --> 0 files

mbx with header "\x4a\x4d\x46\x36" and footer "" --> 0 files
wpc with header "\x3f\x57\x50\x43" and footer "" --> 0 files
htm with header "\x3c\x68\x74\x6d\x6c" and footer "\x3c\x2f\x68\x74\x6d\x6c\x3e" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0d" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0a" --> 0 files
mail with header "\x41\x4f\x4c\x56\x4d" and footer "" --> 0 files
pgd with header "\x50\x47\x50\x64\x4d\x41\x49\x4e\x60\x01" and footer "" --> 0 files
pgp with header "\x99\x00" and footer "" --> 1 files
pgp with header "\x95\x01" and footer "" --> 0 files
pgp with header "\x95\x00" and footer "" --> 2 files
pgp with header "\xa6\x00" and footer "" --> 0 files
txt with header "\x2d\x2d\x2d\x2d\x2d\x42\x45\x47\x49\x4e\x20\x50\x47\x50" and footer "" --> 0 files
rpm with header "\xed\xab" and footer "" --> 0 files
wav with header "\x52\x49\x46\x46\x3f\x3f\x3f\x57\x41\x56\x45" and footer "" --> 0 files
ra with header "\x2e\x72\x61\xfd" and footer "" --> 0 files
ra with header "\x2e\x52\x4d\x46" and footer "" --> 0 files
dat with header "\x72\x65\x67\x66" and footer "" --> 0 files
dat with header "\x43\x52\x45\x47" and footer "" --> 0 files
zip with header "\x50\x4b\x03\x04" and footer "\x3c\xac" --> 0 files
java with header "\xca\xfe\xba\xbe" and footer "" --> 0 files
max with header "\x56\x69\x47\x46\x6b\x1a\x00\x00\x00\x00" and footer "\x00\x00\x05\x80\x00\x00" --> 0 files
pins with header "\x50\x49\x4e\x53\x20\x34\x2e\x32\x30\x0d" and footer "" --> 0 files
Carving files from image.
Image file pass 2/2.
/root/Desktop/test/c.mp3: 100.0% |*****| 18.9 KB 00:00 ETAOPENING
/root/Desktop/test/output_mp3/pgp-34-0/00000003.pgp
CLOSING /root/Desktop/test/output_mp3/pgp-34-0/00000003.pgp
OPENING /root/Desktop/test/output_mp3/pgp-34-0/00000002.pgp
CLOSING /root/Desktop/test/output_mp3/pgp-34-0/00000002.pgp
OPENING /root/Desktop/test/output_mp3/pgp-32-0/00000001.pgp
CLOSING /root/Desktop/test/output_mp3/pgp-32-0/00000001.pgp
OPENING /root/Desktop/test/output_mp3/jpg-4-0/00000000.jpg
CLOSING /root/Desktop/test/output_mp3/jpg-4-0/00000000.jpg
Processing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 4, elapsed = 0 seconds.

Appendix 2: Suspicious_File

1) edit the scalpel.conf file to so that would allow scalpel carve any file type.

```
root@kali:~# vi /etc/scalpel/scalpel.conf
```

```
root@kali:~# scalpel -o /root/Desktop/test -v /root/Desktop/test/Suspicious_File
```

Scalpel version 1.60

Written by Golden G. Richard III, based on Foremost 0.69.

Output directory: "/root/Desktop/test"

Configuration file: "/etc/scalpel/scalpel.conf"

Coverage maps directory: "/root/Desktop/test"

2) root@kali:~# scalpel -o /root/Desktop/test/output/ -v /root/Desktop/test/Suspicious_File

Scalpel version 1.60

Written by Golden G. Richard III, based on Foremost 0.69.

Output directory: "/root/Desktop/test/output"

Configuration file: "/etc/scalpel/scalpel.conf"

Coverage maps directory: "/root/Desktop/test/output"

Opening target "/root/Desktop/test/Suspicious_File"

Total file size is 1304576 bytes

Image file pass 1/2.

Read 1304576 bytes from image file.

```
/root/Desktop/test/Suspicious_File: 100.0% |*****| 1.2 MB 00:00 ETA doc header was found at : 0
```

Memory reallocation performed, total header storage = 101

A doc footer was found at : 0

Memory reallocation performed, total footer storage = 101

A doc header was found at : 0

Memory reallocation performed, total header storage = 101

A pgp header was found at : 64096

Memory reallocation performed, total header storage = 101

A pgp header was found at : 523956

A pgp header was found at : 94893

Memory reallocation performed, total header storage = 101

A pgp header was found at : 95097

A pgp header was found at : 149356

A pgp header was found at : 196176

A pgp header was found at : 262143

A pgp header was found at : 856818

A pgp header was found at : 982885

A pgp header was found at : 1108369

A pgp header was found at : 1204055

A pgp header was found at : 1217347

A pgp header was found at : 1223699

A pgp header was found at : 1237111

A pgp header was found at : 1277062

A pgp header was found at : 18771

Memory reallocation performed, total header storage = 101

A pgp header was found at : 1048235

A pgp header was found at : 1167740

A pgp header was found at : 64148

Memory reallocation performed, total header storage = 101

A pgp header was found at : 87220

A pgp header was found at : 445245

A pgp header was found at : 469416

A pgp header was found at : 563261

A pgp header was found at : 648687
A pgp header was found at : 654045
A pgp header was found at : 657066
A pgp header was found at : 905029
A pgp header was found at : 909748
A pgp header was found at : 934739
A pgp header was found at : 937332
A pgp header was found at : 939828
A pgp header was found at : 949423

Allocating work queues...

Work queues allocation complete. Building carve lists...

Carve lists built. Workload:

art with header "\x4a\x47\x04\x0e" and footer "\xcf\xc7\xcb" --> 0 files
art with header "\x4a\x47\x03\x0e" and footer "\xd0\xcb\x00\x00" --> 0 files
gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" --> 0 files
gif with header "\x47\x49\x46\x38\x39\x61" and footer "\x00\x3b" --> 0 files
jpg with header "\xff\xd8\xff\xe0\x00\x10" and footer "\xff\xd9" --> 0 files
png with header "\x50\x4e\x47\x3f" and footer "\xff\xfc\xfd\xfe" --> 0 files
bmp with header "\x42\x4d\x3f\x3f\x00\x00\x00" and footer "" --> 0 files
tif with header "\x49\x49\x2a\x00" and footer "" --> 0 files
tif with header "\x4d\x4d\x00\x2a" and footer "" --> 0 files

.....(NOTE: we removed pgp files due to redundancy).....

doc with header "\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00"
and footer "\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00" --> 1 files
doc with header "\xd0\xcf\x11\xe0\xa1\xb1" and footer "" --> 1 files
pst with header "\x21\x42\x4e\x5a\x6f\xb5\x6a" and footer "" --> 0 files
dbx with header "\xcf\xad\x12\xfe\xc5\xfd\x74\x6f" and footer "" --> 0 files
idx with header "\x4a\x4d\x46\x39" and footer "" --> 0 files
mbx with header "\x4a\x4d\x46\x36" and footer "" --> 0 files
wpc with header "\x3f\x57\x50\x43" and footer "" --> 0 files
htm with header "\x3c\x68\x74\x6d\x6c" and footer "\x3c\x2f\x68\x74\x6d\x6c\x3e" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0d" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x4f\x46\x0a" --> 0 files
mail with header "\x41\x4f\x4c\x56\x4d" and footer "" --> 0 files
pgd with header "\x50\x47\x50\x64\x4d\x41\x49\x4e\x60\x01" and footer "" --> 0 files
pgp with header "\x99\x00" and footer "" --> 2 files
pgp with header "\x95\x01" and footer "" --> 13 files
pgp with header "\x95\x00" and footer "" --> 656 files
pgp with header "\xa6\x00" and footer "" --> 14 files
txt with header "\x2d\x2d\x2d\x2d\x2d\x42\x45\x47\x49\x4e\x20\x50\x47\x50" and footer "" --> 0 files
rpm with header "\xed\xab" and footer "" --> 0 files
wav with header "\x52\x49\x46\x46\x3f\x3f\x3f\x57\x41\x56\x45" and footer "" --> 0 files
ra with header "\x2e\x72\x61\xfd" and footer "" --> 0 files
ra with header "\x2e\x52\x4d\x46" and footer "" --> 0 files
dat with header "\x72\x65\x67\x66" and footer "" --> 0 files
dat with header "\x43\x52\x45\x47" and footer "" --> 0 files
zip with header "\x50\x4b\x03\x04" and footer "\x3c\xac" --> 0 files
java with header "\xca\xfe\xba\xbe" and footer "" --> 0 files
max with header "\x56\x69\x47\x46\x6b\x1a\x00\x00\x00" and footer "\x00\x00\x05\x80\x00\x00" --> 0 files
pins with header "\x50\x49\x4e\x53\x20\x34\x2e\x32\x30\x0d" and footer "" --> 0 files

Carving files from image.

Image file pass 2/2.

/root/Desktop/test/Suspicious_File: 100.0% |*****|

1.2 MB 00:00 ETAOPENING /root/Desktop/test/output/pgp-35-0/00000686.pgp

CLOSING /root/Desktop/test/output/pgp-35-0/00000686.pgp

```
OPENING /root/Desktop/test/output/pgp-35-0/00000685.pgp
CLOSING /root/Desktop/test/output/pgp-35-0/00000685.pgp
OPENING /root/Desktop/test/output/pgp-35-0/00000684.pgp
CLOSING /root/Desktop/test/output/pgp-35-0/00000684.pgp
```

.....(nOTE: we removed pgp files due to redundancy).....

```
OPENING /root/Desktop/test/output/pgp-33-0/00000010.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000010.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000009.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000009.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000008.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000008.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000007.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000007.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000006.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000006.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000005.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000005.pgp
OPENING /root/Desktop/test/output/pgp-33-0/00000004.pgp
CLOSING /root/Desktop/test/output/pgp-33-0/00000004.pgp
OPENING /root/Desktop/test/output/pgp-32-0/00000003.pgp
CLOSING /root/Desktop/test/output/pgp-32-0/00000003.pgp
OPENING /root/Desktop/test/output/pgp-32-0/00000002.pgp
CLOSING /root/Desktop/test/output/pgp-32-0/00000002.pgp
OPENING /root/Desktop/test/output/doc-21-0/00000001.doc
CLOSING /root/Desktop/test/output/doc-21-0/00000001.doc
OPENING /root/Desktop/test/output/doc-20-0/00000000.doc
CLOSING /root/Desktop/test/output/doc-20-0/00000000.doc
```

Processing of image file complete. Cleaning up...

Done.

Scalpel is done, files carved = 687, elapsed = 1 seconds.

References

- 1) <http://md5deep.sourceforge.net/>
- 2) www.md5deep.sourceforge.net/md5deep.html
- 3) http://www.garykessler.net/library/file_sigs.html
- 4) <http://araskin.webs.com/exif/exif.html>