Digital Forensics and Cyber Security Tools Final Submission Portfolio

Thomas MacKinnon TP066728
Intake Code: APDMF2204CYS(PR)
Module Code: CT122-0-M-FST-T-1
Dr. Mohamed Shabbir Hamza Abdulnabi
Word Count: 3211

Contents

List of Figures

List of Tables

1 Introduction

The incorporation of technology into our everyday lives has led to a new world of possibilities at the public's' finger tips, with great advances in productivity and communication worldwide. This great leap has of course come with consequences, Cyber crime has risen at an alarming rate, and has no indication of slowing down anytime soon. The FBI (Federal Bureau of Investigation) released the annual IC3 report (Internet Crime Complaint Centre) in 2020, which detailed the shocking rise of Internet crime reports, increasing in more than 300,000 cases, as seen in figure 1.

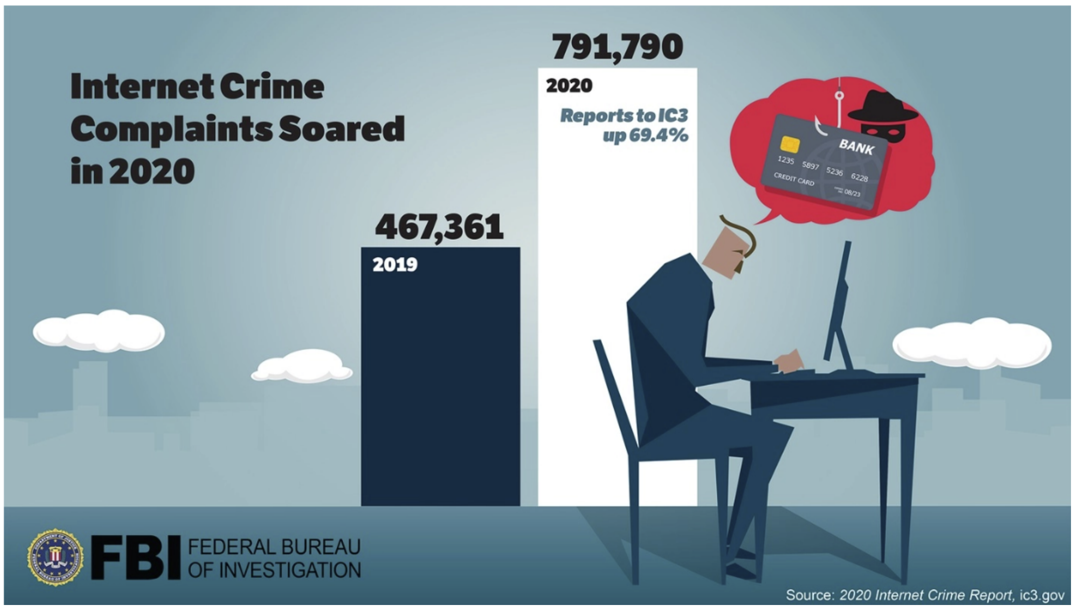


Figure 1: IC3 reported cases from 2020 compared to 2019 (FBI, 2021)

The report estimates that $4.2 Billion was lost to Internet Crime in 2020 (FBI, 2020), and of course this figure will keep rising higher and higher. From this a bustling industry of Security Researchers have emerged, with many different areas to specialise in. Some examples are Penetration testers, who vigorously test a businesses' security with the intentions of finding weak points before a malicious hacker can, and a Digital Forensics Investigator, who aims to uncover evidence left behind on a suspects' devices. The common link between all these professions is the use of tools to aid in their Cyber Security and Digital Forensic Investigations, taking the form of specialised software (Al-Matari et al, 2018). Tools are essential to any successful investigation, and can perform a huge variety of functions, from password cracking to network reconnaissance, it is vital that any Cyber Security specialist is familiar with the tools of the trade.

This report aims to give an overview of everyday tools used in Digital Forensics and Cyber Security cases, by explaining their function, results and how they can be used in an investigation. This will be done through a step by step utilising screenshots to properly display the use of the tool. The Report will end on a case study investigation followed by a conclusion of the findings.

2 Portfolio 1: Hex Editor, Wireshark, and Exif Tool

2.1 Hex Editor

2.1.1 Introduction

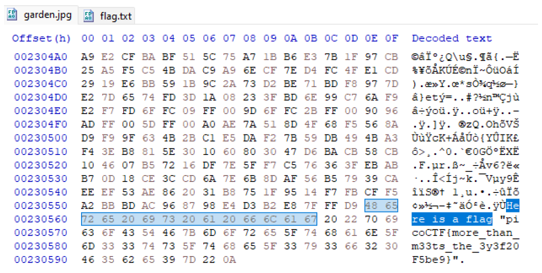
A Hex Editor is tool used to view and edit binary data of a file, presenting the data in hexadecimal format. Hex Editors are a useful tool in Digital Forensics and Cyber Security Investigations as they allow the user the verify the file type authenticity by checking the file signatures. Malicious actors can easily change the file type to something else in an attempt to hide the true contents, by using a Hex Editor the true file type can be easily found out by cross referencing the file signature with an online database. It is beneficial to be able to recognise file signatures, so a table of common file types with signatures has been constructed, as seen in Table 1.

Hex Editor Results	
File Format	File Signature
JPEG (.jpg)	FF D8 FF E0
PNG (.png)	89 50 4E 47 0D 0A 1A 0A
Word Document (.docx)	50 4B 03 04
Word Document (.doc)	D0 CF 11 E0 A1 B1 1A E1
PDF Document (.pdf)	25 50 44 46
Excel Spreadsheet (.xls)	D0 CF 11 E0 A1 B1 1A E1
Excel Spreadsheet (.xlsx)	50 4B 03 04
Powerpoint (.ppt)	D0 CF 11 E0 A1 B1 1A E1
Powerpoint (.pptx)	50 4B 03 04
Executable File (.exe)	4D 5A
MP3 File (.mp3)	49 44 33

Table 1: Table displaying File Signatures for popular File types

2.1.2 Hidden data in garden.jpg

A file was provided, named "garden.jpg", that was suspected of having hidden content on it. The image was opened in HxD, a free Hex Editing software, and the file signature was verified using Table 1. The file was checked over, and was found to contain hidden text after the end of the file, as seen Figure 2, which read "Here is a flag". If this had been a real scenario the "search/find" feature in the software would have been used to search for important phrases like passwords or links.

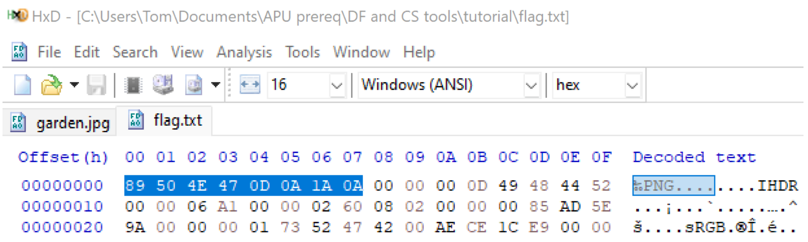


```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

002304A0  A9 E2 CF BA BF 51 5C 75 A7 1B B6 E3 7B 1F 97 CB  ©âÏº¿Q\u§.¶ã{.—Ë
002304B0  25 A5 F5 C5 4B DA C9 A9 6E CF 7E D4 FC 4F E1 CD  %¥õÅKÚÉ©nÏ~ÔüOáÍ
002304C0  29 19 E6 BB 59 1B 9C 2A 73 D2 BE 71 BD F8 97 7D  ).æ»Y.œ*sÒ¾q½ø—}
002304D0  E2 7D 65 74 FD 3D 1A 08 23 3F BD 6E 99 C7 6A F9  â}etý=..#?½n™Çjù
002304E0  E2 F7 FD 6F FC 09 FF 00 9D 6F FC 2B FF 00 90 96  â÷ýoü.ÿ..oü+ÿ..–
002304F0  AD FF 00 5D FF 00 A0 AE 7A 51 8D 4F 68 F5 56 8A  .ÿ.]ÿ. ®zQ.OhõVŠ
00230500  D9 F9 9F 63 4B 2B C1 E5 DA F2 7B 59 DB 49 4B A3  ÙùŸcK+ÁåÚò{YÛIK£
00230510  F4 3E B8 81 5E 30 10 60 80 30 47 D6 BA CB 58 CB  ô>¸.^0.`€0GÖºËXË
00230520  10 46 07 B5 72 16 DF 7E 5F F7 C5 76 36 3F EB AB  .F.µr.ß~_÷Åv6?ë«
00230530  B7 0D 18 CE 3C CD 6A 7E 6B 8D AF 56 B5 79 39 CA  ·..Î<Íj~k.¯Vµy9Ê
00230540  EE EF 53 AE 86 20 31 B8 75 1F 95 14 F7 FB CF F5  îïS®† 1¸u.•.÷ûÏõ
00230550  A2 BB BD AC 96 87 98 E4 D3 B2 E8 7F FF D9 48 65  ¢»½¬–‡˜äÓ²è.ÿÙHe
00230560  72 65 20 69 73 20 61 20 66 6C 61 67 20 22 70 69  re is a flag "pi
00230570  63 6F 43 54 46 7B 6D 6F 72 65 5F 74 68 61 6E 5F  coCTF{more_than_
00230580  6D 33 33 74 73 5F 74 68 65 5F 33 79 33 66 32 30  m33ts_the_3y3f20
00230590  46 35 62 65 39 7D 22 0A                          F5be9}".
```

Figure 2: Hidden Text within "garden.jpg"

2.1.3 Hidden Flag in flag.txt

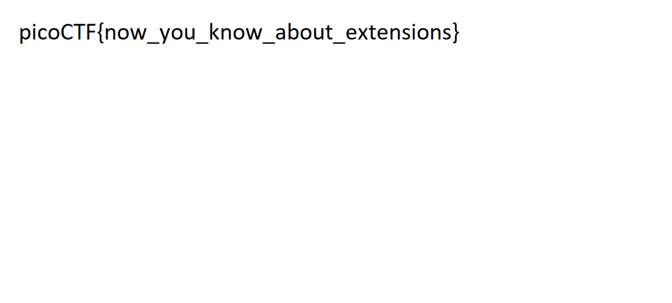

HxD - [C:\Users\Tom\Documents\APU prereq\DF and CS tools\tutorial\flag.txt]
File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
garden.jpg
flag.txt

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010  00 00 06 A1 00 00 02 60 08 02 00 00 00 85 AD 5E  ...¡...`.......^
00000020  9A 00 00 00 01 73 52 47 42 00 AE CE 1C E9 00 00  š....sRGB.®Î.é..
```

Figure 3: Wrong File format for “flag.txt”

Another file was provided, named “flag.txt”, and was opened with HxD. When consulting Table 1 it became apparent that “flag.txt” was not a Text file but in fact a PNG file from its file signature, as seen in Figure 3. The file was changed to a PNG file type by renaming it, which revealed the image in Figure 4, and revealing the Flag.



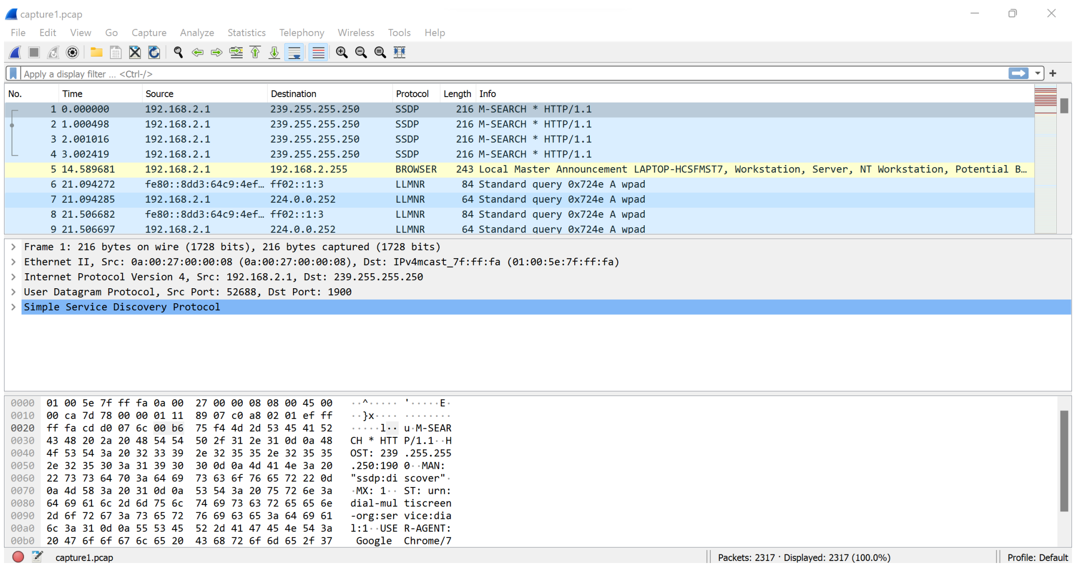
picoCTF{now_you_know_about_extensions}

Figure 4: Hidden flag within “flag.txt” after being changed to a PNG file

2.2 Wireshark

Wireshark is a packet analyser tool used commonly in Network Security and Digital Forensics cases. The program allows for live capture of network traffic, which can easily be saved to a PCAP file (packet capture, .pcap) for later analysis. Wireshark also provides a variety of filters to limit results to that of only one host, or specific protocols, allowing for more efficient investigation. A noteable feature is the “Follow” option, that allows the user to view a conversation between two or more computers through a series of protocols (TCP, UDP, HTTP). In many Digital Forensics cases Investigators can find useful or incriminating evidence from network captures, making it a cornerstone of the investigation process.

A PCAP file was provided, named “capture1.pcap”, with a hidden flag somewhere inside it. To start, open up the file in Wireshark, the file contents can look intimidating at first it is really quite simple. Figure 5 shows the contents of “capture1.pcap”, with the tool bar at the top, followed by the display filter search bar, a column displaying each packets information, followed by the frame details of the selected packet, and lastly a Hexadecimal display of the packet contents.


```
No.  Time       Source                  Destination      Protocol  Length Info
  1 0.000000    192.168.2.1             239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
  2 1.000498    192.168.2.1             239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
  3 2.001016    192.168.2.1             239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
  4 3.002419    192.168.2.1             239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
  5 14.589681   192.168.2.1             192.168.2.255    BROWSER   243 Local Master Announcement LAPTOP-HCSFMST7, Workstation, Server, NT Workstation, Potential B…
  6 21.094272   fe80::8dd3:64c9:4ef…    ff02::1:3        LLMNR      84 Standard query 0x724e A wpad
  7 21.094285   192.168.2.1             224.0.0.252      LLMNR      64 Standard query 0x724e A wpad
  8 21.506682   fe80::8dd3:64c9:4ef…    ff02::1:3        LLMNR      84 Standard query 0x724e A wpad
  9 21.506697   192.168.2.1             224.0.0.252      LLMNR      64 Standard query 0x724e A wpad
```

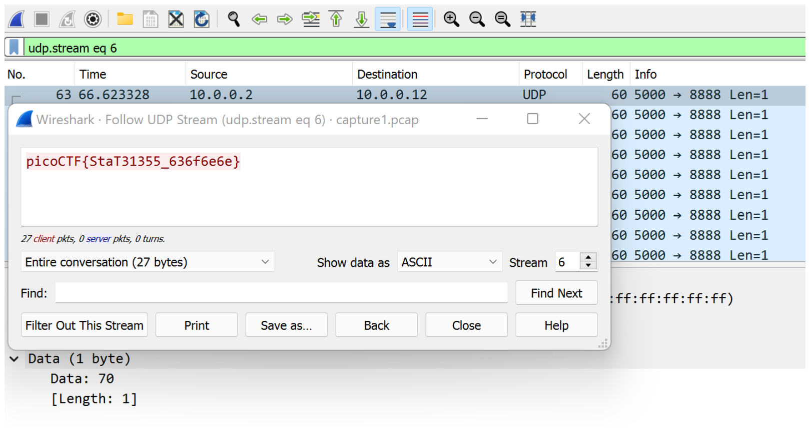
```
> Frame 1: 216 bytes on wire (1728 bits), 216 bytes captured (1728 bits)
> Ethernet II, Src: 0a:00:27:00:00:08 (0a:00:27:00:00:08), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
> Internet Protocol Version 4, Src: 192.168.2.1, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 52688, Dst Port: 1900
> Simple Service Discovery Protocol
```

```
0000  01 00 5e 7f ff fa 0a 00  27 00 00 08 08 00 45 00   ··^····· '·····E·
0010  00 ca 7d 78 00 00 01 11  89 07 c0 a8 02 01 ef ff   ··}x···· ········
0020  ff fa cd d0 07 6c 00 b6  75 f4 4d 2d 53 45 41 52   ·····l·· u·M-SEAR
0030  43 48 20 2a 20 48 54 54  50 2f 31 2e 31 0d 0a 48   CH * HTT P/1.1··H
0040  4f 53 54 3a 20 32 33 39  2e 32 35 35 2e 32 35 35   OST: 239 .255.255
0050  2e 32 35 30 3a 31 39 30  30 0d 0a 4d 41 4e 3a 20   .250:190 0··MAN:
0060  22 73 73 64 70 3a 64 69  73 63 6f 76 65 72 22 0d   "ssdp:di scover"·
0070  0a 4d 58 3a 20 31 0d 0a  53 54 3a 20 75 72 6e 3a   ·MX: 1·· ST: urn:
0080  64 69 61 6c 2d 6d 75 6c  74 69 73 63 72 65 65 6e   dial-mul tiscreen
0090  2d 6f 72 67 3a 73 65 72  76 69 63 65 3a 64 69 61   -org:ser vice:dia
00a0  6c 3a 31 0d 0a 55 53 45  52 2d 41 47 45 4e 54 3a   l:1··USE R-AGENT:
00b0  20 47 6f 6f 67 6c 65 20  43 68 72 6f 6d 65 2f 37    Google  Chrome/7
```

capture1.pcap | Packets: 2317 · Displayed: 2317 (100.0%) | Profile: Default

Figure 5: Wireshark showing network traffic data for capture1.pcap

A good place to look for a flag is in the conversations between two or more computers, by going to the "Analyze" option on the toolbar and hovering over "Follow" it is possible to see any mediums that were used to communicate. This reveals that the a User Data Protocol (UDP) conversation happened, after selecting that option a series of streams can be viewed. Scrolling through these streams reveals the flag at stream 6, as seen in Figure 6.


Filter: udp.stream eq 6

```
No.  Time       Source      Destination  Protocol  Length Info
 63 66.623328   10.0.0.2    10.0.0.12    UDP       60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
                                                   60 5000 → 8888 Len=1
```

Wireshark · Follow UDP Stream (udp.stream eq 6) · capture1.pcap

```
picoCTF{StaT31355_636f6e6e}
```

27 client pkts, 0 server pkts, 0 turns.

Entire conversation (27 bytes) | Show data as ASCII | Stream 6

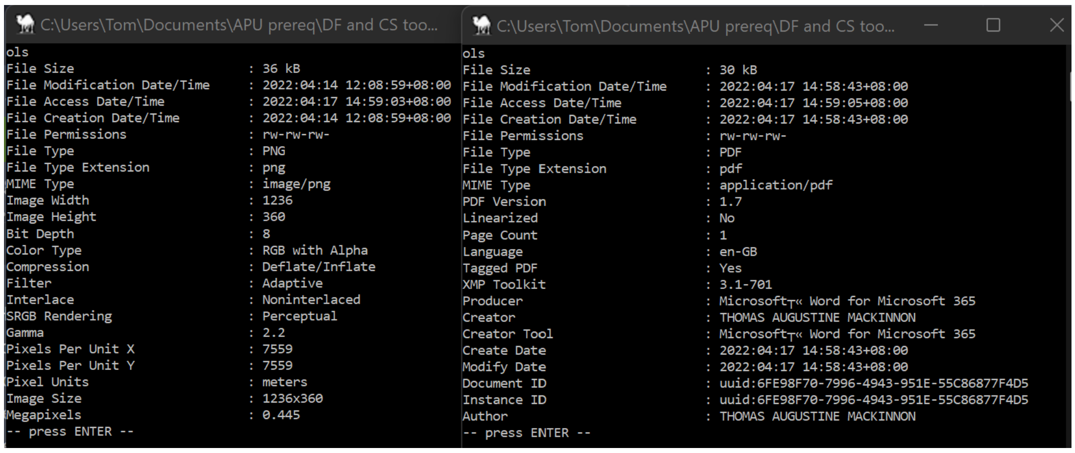
```
:ff:ff:ff:ff:ff)
Data (1 byte)
    Data: 70
    [Length: 1]
```

Figure 6: Wireshark displaying the found flag in the sixth UDP stream

2.3 Exif Tool

Metadata is a type of digital information that gives details about other data, such as for an image Metadata contain the resolution, file size, and date created. Metadata can seem inconsequential at first, but offers a lot with data such a geo-location, which could link a suspect to the crime scene and become invaluable evidence for a digital forensics case. This scenario actually happened when Vice, the alternative news publication, released a photo onto their twitter boasting that they were with John McAfee, the creator of the McAfee Antivirus software. McAfee at the time was wanted for murder and had escaped to Guatemala illegally, Vice had not removed the Metadata of the Image so had unknowingly revealed his exact location (ODS, 2017). Figure 7 shows an example of a popular tool named "ExifTool" displaying the metadata of a PNG image and a PDF file. Using ExifTool in this way reveals useful information such as the original author of the File, and what software the file was created in.

ExifTool is easy to use and is on multiple operating systems, and can check the Metadata of Image, Audio, Video, and PDF files. In Windows, simply drag the file onto the ExifTool icon and the information window will pop up in a Windows Terminal. In Linux, simply use the command `exiftool file.extension` in the Linux Command Line.

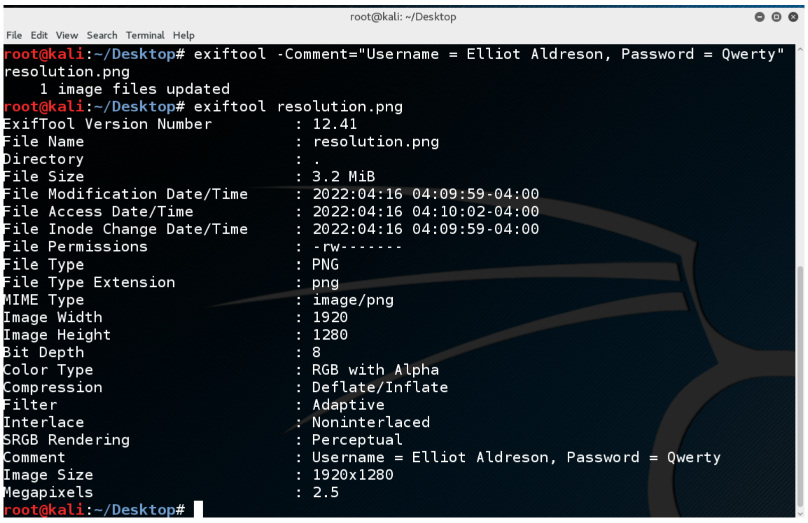

C:\Users\Tom\Documents\APU prereq\DF and CS too...
C:\Users\Tom\Documents\APU prereq\DF and CS too...

```
ols
File Size                       : 36 kB
File Modification Date/Time     : 2022:04:14 12:08:59+08:00
File Access Date/Time           : 2022:04:17 14:59:03+08:00
File Creation Date/Time         : 2022:04:14 12:08:59+08:00
File Permissions                : rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1236
Image Height                    : 360
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Gamma                           : 2.2
Pixels Per Unit X               : 7559
Pixels Per Unit Y               : 7559
Pixel Units                     : meters
Image Size                      : 1236x360
Megapixels                      : 0.445
-- press ENTER --
```

```
ols
File Size                       : 30 kB
File Modification Date/Time     : 2022:04:17 14:58:43+08:00
File Access Date/Time           : 2022:04:17 14:59:05+08:00
File Creation Date/Time         : 2022:04:17 14:58:43+08:00
File Permissions                : rw-rw-rw-
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.7
Linearized                      : No
Page Count                      : 1
Language                        : en-GB
Tagged PDF                      : Yes
XMP Toolkit                     : 3.1-701
Producer                        : Microsoft┬« Word for Microsoft 365
Creator                         : THOMAS AUGUSTINE MACKINNON
Creator Tool                    : Microsoft┬« Word for Microsoft 365
Create Date                     : 2022:04:17 14:58:43+08:00
Modify Date                     : 2022:04:17 14:58:43+08:00
Document ID                     : uuid:6FE98F70-7996-4943-951E-55C86877F4D5
Instance ID                     : uuid:6FE98F70-7996-4943-951E-55C86877F4D5
Author                          : THOMAS AUGUSTINE MACKINNON
-- press ENTER --
```

Figure 7: ExifTool working on a .png and .pdf File

ExifTool can also be used to edit or remove Metadata content, which is particularly useful for a malicious hacker wanting to cover their tracks. This feature can also be a good way to hide or transfer important information that is sensitive in nature. To demonstrate this ExifTool was run in the Linux Command line and the command seen in Figure 8 was used to add a Comment to the Metadata with login information. This Command can easily be edited to add other data such as a new author or location simply by changing `-Comment=` to something like `-Location="North Pole"`.



```
root@kali:~/Desktop# exiftool -Comment="Username = Elliot Aldreson, Password = Qwerty"
resolution.png
    1 image files updated
root@kali:~/Desktop# exiftool resolution.png
ExifTool Version Number         : 12.41
File Name                       : resolution.png
Directory                       : .
File Size                       : 3.2 MiB
File Modification Date/Time     : 2022:04:16 04:09:59-04:00
File Access Date/Time           : 2022:04:16 04:10:02-04:00
File Inode Change Date/Time     : 2022:04:16 04:09:59-04:00
File Permissions                : -rw-------
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1920
Image Height                    : 1280
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Comment                         : Username = Elliot Aldreson, Password = Qwerty
Image Size                      : 1920x1280
Megapixels                      : 2.5
root@kali:~/Desktop#
```

Figure 8: ExifTool used to add a comment to an Image file

2.4 Capture the flag Challenge

To further explore and develop skill in Digital Forensics and Cyber Security Tools a challenge was set to create a mini Capture the Flag (CTF) for a classmate. After creation and giving the challenge out the investigator would receive a classmate's challenge to try and solve.

2.4.1 Creation of CTF challenge

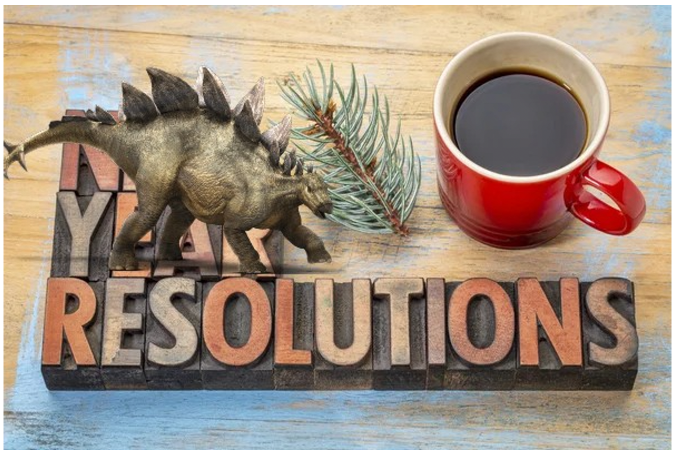
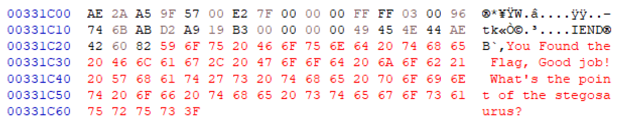


Figure 9: Image created with Pixlr named "resolution.png" with hints for the challenge

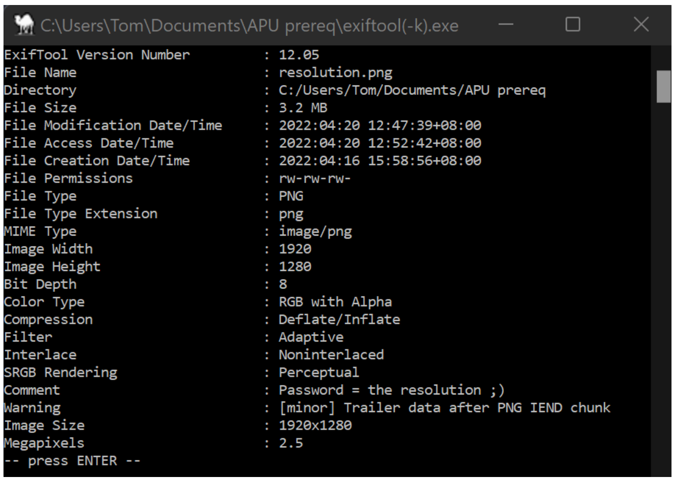
To start the challenge an image was created in Pixlr, an online photo editing tool, which can be seen in Figure 9. The image was deliberately edited to have a Stegosaurus covering text, hinting that Steganography might be involved later in the challenge, and that the word "resolution" might be important. The image was created with a large resolution and high quality option selected so it would be big enough to hide a file in later on. The image was opened with the HxD hex editing tool used earlier, and scrolled to the bottom of its contents to add a piece of text. The text, which can be seen in Figure 10, as is meant to serve as a red herring from the true flag.



```
00331C00  AE 2A A5 9F 57 00 E2 7F 00 00 00 FF FF 03 00 96  ®*¥ŸW.â....ÿÿ..–
00331C10  74 6B AB D2 A9 19 B3 00 00 00 00 49 45 4E 44 AE  tk«Ò©.³....IEND®
00331C20  42 60 82 59 6F 75 20 46 6F 75 6E 64 20 74 68 65  B`,You Found the
00331C30  20 46 6C 61 67 2C 20 47 6F 6F 64 20 6A 6F 62 21   Flag, Good job!
00331C40  20 57 68 61 74 27 73 20 74 68 65 20 70 6F 69 6E   What's the poin
00331C50  74 20 6F 66 20 74 68 65 20 73 74 65 67 6F 73 61  t of the stegosa
00331C60  75 72 75 73 3F                                   urus?
```

Figure 10: Fake flag added to "resolution.jpg" in HxD

If the investigator is curious enough they will find that the metadata for the image has been edited, now including a hint to a password as seen in Figure 11. The resolution, or image size as ExifTool puts it, is "1920x1280", so would be the hidden password. This was achieved using ExifTool in Linux Command line and using the command `exiftool -Comment="Password = the resolution ;)" resolution.png`.

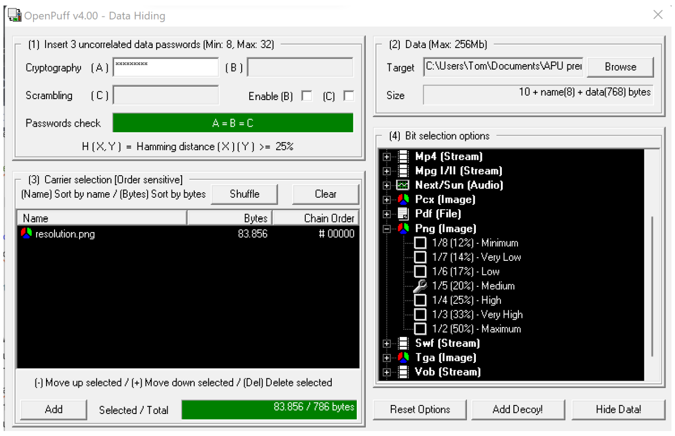


C:\Users\Tom\Documents\APU prereq\exiftool(-k).exe

```
ExifTool Version Number         : 12.05
File Name                       : resolution.png
Directory                       : C:/Users/Tom/Documents/APU prereq
File Size                       : 3.2 MB
File Modification Date/Time     : 2022:04:20 12:47:39+08:00
File Access Date/Time           : 2022:04:20 12:52:42+08:00
File Creation Date/Time         : 2022:04:16 15:58:56+08:00
File Permissions                : rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1920
Image Height                    : 1280
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Comment                         : Password = the resolution ;)
Warning                         : [minor] Trailer data after PNG IEND chunk
Image Size                      : 1920x1280
Megapixels                      : 2.5
-- press ENTER --
```

Figure 11: Exiftool revealing the hidden hint for a password

Steganography was used to hide the final true flag, seen Figure 13, which is a simple ASCII flag in a text document. OpenPuff, a free Steganography decoder/encoder, was used to hide the text file within "resolution.png". From the start menu the "Hide" option was selected, and the option to have multiple passwords was disabled (tick boxes named "Enable (B)" and "(c)"), as to not make it too hard. The password was set as "1920x1280", the image resolution, and a target file was set as "flag.txt" in the second option tile. The Carrier was set as "resolution.txt" in third option tile, and the bit selection option was set to "PNG(Image)" and then to the medium option. The "Hide Data!" option was selected and the CTF was finished, the full layout and selected options for OpenPuff can be seen in Figure 12.



OpenPuff v4.00 - Data Hiding

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) ********* (B)

Scrambling (C) Enable (B) ☐ (C) ☐

Passwords check: A = B = C

H (X, Y) = Hamming distance (X) (Y) >= 25%

(2) Data (Max: 256Mb)

Target: C:\Users\Tom\Documents\APU pre

Size: 10 + name(8) + data(768) bytes

(3) Carrier selection [Order sensitive]

(Name) Sort by name / (Bytes) Sort by bytes

Name	Bytes	Chain Order
resolution.png	83.856	# 00000

(-) Move up selected / (+) Move down selected / (Del) Delete selected

Selected / Total: 83.856 / 786 bytes

(4) Bit selection options

- Mp4 (Stream)
- Mpg I/II (Stream)
- Next/Sun (Audio)
- Pcx (Image)
- Pdf (File)
- Png (Image)
  - 1/8 (12%) - Minimum
  - 1/7 (14%) - Very Low
  - 1/6 (17%) - Low
  - 1/5 (20%) - Medium ✓
  - 1/4 (25%) - High
  - 1/3 (33%) - Very High
  - 1/2 (50%) - Maximum
- Swf (Stream)
- Tga (Image)
- Vob (Stream)

Figure 12: OpenPuff options for “resolution.png”

To test this works the “Unhide” option was selected in OpenPuff, and the password was entered with the right file. This produced the “flag.txt” file into whatever the chosen directory was proving it could be found.

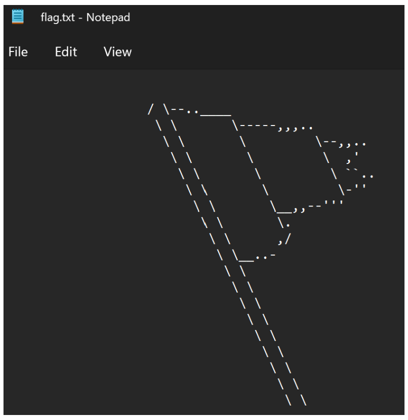
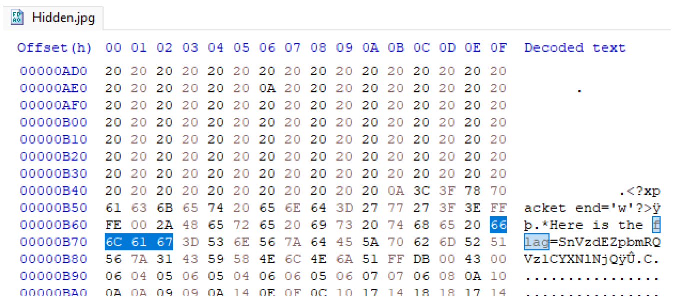


Figure 13: Real flag, being an ASCII text file, hidden inside “resolution.jpg”

2.4.2 Solution of Classmate's CTF challenge

Figure 14: Image for "Hidden.png" for the CTF challenge

A Capture the Flag challenge was assigned by a fellow student, being Soh Yoke Cheng (TP047206). An image named "Hidden.png" was provided, as seen in Figure 14 to investigate. The image was opened up in HxD to check the hexdata, which revealed that the file signature was for a JPG and not a PNG, so the extension was changed accordingly. Further search was conducted for hidden text, nothing was obviously hidden, until a string search for "flag" revealed the contents of Figure 15. The flag, being "SnVzdEZpbmRQVz1CYXNlNjQ", seemed to be encrypted, so CyberChef (GCHQ, N.D) was used to decrypt it. CyberChef spotted that it was encoded in Base64, so after decrypting it the flag was "JustFindPW=Base64". This seemed like a fake flag, hinting at a Password hidden somewhere.



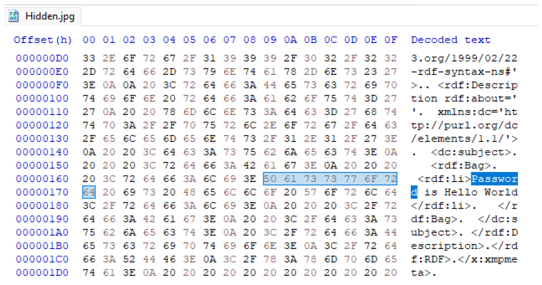
```
Hidden.jpg

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000AD0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000AE0  20 20 20 20 20 20 0A 20 20 20 20 20 20 20 20 20        .
00000AF0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000B00  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000B10  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000B20  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000B30  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000B40  20 20 20 20 20 20 20 20 20 20 20 0A 3C 3F 78 70             .<?xp
00000B50  61 63 6B 65 74 20 65 6E 64 3D 27 77 27 3F 3E FF  acket end='w'?>ÿ
00000B60  FE 00 2A 48 65 72 65 20 69 73 20 74 68 65 20 66  þ.*Here is the f
00000B70  6C 61 67 3D 53 6E 56 7A 64 45 5A 70 62 6D 52 51  lag=SnVzdEZpbmRQ
00000B80  56 7A 31 43 59 58 4E 6C 4E 6A 51 FF DB 00 43 00  Vz1CYXNlNjQÿÛ.C.
00000B90  06 04 05 06 05 04 06 06 05 06 07 07 06 08 0A 10  ................
00000BA0  0A 0A 09 09 0A 14 0E 0F 0C 10 17 14 18 18 17 14  ................
```

Figure 15: HxD revealing an encoded flag in "Hidden.jpg"

Another string search was performed with "password", which revealed the text seen in Figure 16, suggesting that "Hello World" is the password needed to find the flag.



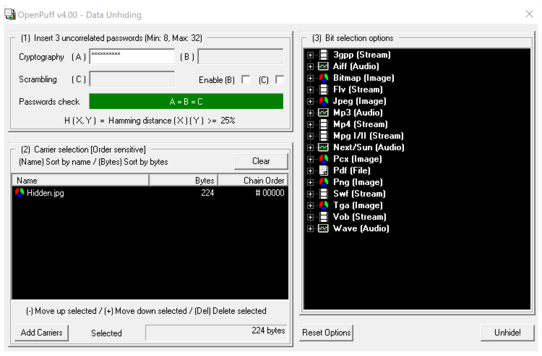
Hidden.jpg

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

000000D0  33 2E 6F 72 67 2F 31 39 39 39 2F 30 32 2F 32 32  3.org/1999/02/22
000000E0  2D 72 64 66 2D 73 79 6E 74 61 78 2D 6E 73 23 27  -rdf-syntax-ns#'
000000F0  3E 0A 0A 20 3C 72 64 66 3A 44 65 73 63 72 69 70  >.. <rdf:Descrip
00000100  74 69 6F 6E 20 72 64 66 3A 61 62 6F 75 74 3D 27  tion rdf:about='
00000110  27 0A 20 20 78 6D 6C 6E 73 3A 64 63 3D 27 68 74  '.  xmlns:dc='ht
00000120  74 70 3A 2F 2F 70 75 72 6C 2E 6F 72 67 2F 64 63  tp://purl.org/dc
00000130  2F 65 6C 65 6D 65 6E 74 73 2F 31 2E 31 2F 27 3E  /elements/1.1/'>
00000140  0A 20 20 3C 64 63 3A 73 75 62 6A 65 63 74 3E 0A  .  <dc:subject>.
00000150  20 20 20 3C 72 64 66 3A 42 61 67 3E 0A 20 20 20     <rdf:Bag>.
00000160  20 3C 72 64 66 3A 6C 69 3E 50 61 73 73 77 6F 72   <rdf:li>Passwor
00000170  64 20 69 73 20 48 65 6C 6C 6F 20 57 6F 72 6C 64  d is Hello World
00000180  3C 2F 72 64 66 3A 6C 69 3E 0A 20 20 20 3C 2F 72  </rdf:li>.   </r
00000190  64 66 3A 42 61 67 3E 0A 20 20 3C 2F 64 63 3A 73  df:Bag>.  </dc:s
000001A0  75 62 6A 65 63 74 3E 0A 20 3C 2F 72 64 66 3A 44  ubject>. </rdf:D
000001B0  65 73 63 72 69 70 74 69 6F 6E 3E 0A 3C 2F 72 64  escription>.</rd
000001C0  66 3A 52 44 46 3E 0A 3C 2F 78 3A 78 6D 70 6D 65  f:RDF>.</x:xmpme
000001D0  74 61 3E 0A 20 20 20 20 20 20 20 20 20 20 20 20  ta>.
```

Figure 16: HxD revealing an password in “Hidden.jpg”

The Password was suspected to be used in a Steganography tool to reveal a hidden text file within the image. OpenPuff was opened and the “Unhide” option was selected. The Password found in the hexdata was entered (Disabling the option for multiple passwords) and “Hidden.jpg” was attached as the carrier, as seen in Figure 17.



OpenPuff v4.00 - Data Unhiding

(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)

Cryptography (A) ********** (B)

Scrambling (C) Enable (B) ☐ (C) ☐

Passwords check: A = B = C

H (X, Y) = Hamming distance (X) (Y) >= 25%

(2) Carrier selection (Order sensitive)

(Name) Sort by name / (Bytes) Sort by bytes

Name	Bytes	Chain Order
Hidden.jpg	224	# 00000

(-) Move up selected / (+) Move down selected / (Del) Delete selected

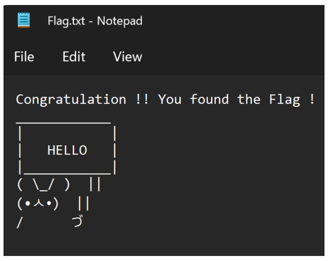
Selected: 224 bytes

(3) Bit selection options

3gpp (Stream)
Aiff (Audio)
Bitmap (Image)
Flv (Stream)
Jpeg (Image)
Mp3 (Audio)
Mp4 (Stream)
Mpg I/II (Stream)
Next/Sun (Audio)
Pcx (Image)
Pdf (File)
Png (Image)
Swf (Stream)
Tga (Image)
Vob (Stream)
Wave (Audio)

Figure 17: “Hidden.jpg” being added to OpenPuff to check for hidden file within

This completed the challenge and revealed the flag, as seen in Figure 18.



Flag.txt - Notepad

File Edit View

```
Congratulation !! You found the Flag !

 ____________
|            |
|   HELLO    |
|____________|
( \_/ )  ||
(•ㅅ•)  ||
/     づ
```

Figure 18: Final Flag found for “Hidden.jpg”

3 Portfolio 2: Imaging tools

Imaging is a core part of any Digital Forensics Investigation, without it evidence would be unusable in court. This is due to the important of evidence integrity, Investigators have to verify that they have made no changes to evidence to make valid for court. Imaging allows Investigators to create a copy of storage devices, such as hard discs or USBs, which can be thoroughly checked through whilst the original evidence is locked away safety. The use of Imaging provides a safe environment for investigating, so learning how to create an Image using tools is a very important step.

3.1 FTK Imager

FTK Imager is an imaging tool that allows for creation and investigation of forensic images. The tool also provides the user with the ability to see deleted files left on storage devices, making it very useful to Investigators (CloudNine, N.D.). The tool was run using the Windows, which brings up an empty screen since there is no evidence yet. A good way to learn FTK Imager is through imaging a USB, which can be done quickly due to the low storage compared to a hard drive.

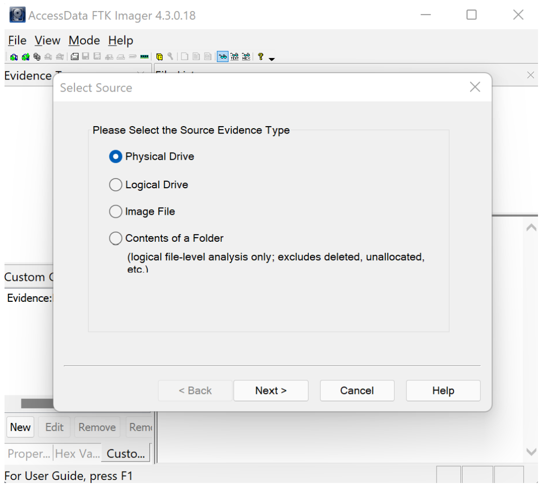
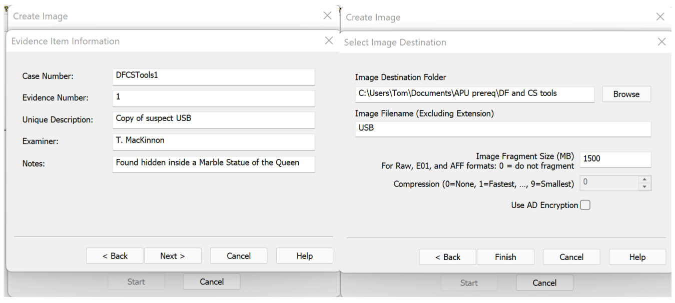


Figure 19: FTK Imager options for adding evidence

To do this select “File” option and then the “Create Disk Image”, which will bring up the option screen seen in Figure 19. As a USB is a Physical drive select the “Physical Drive” option, and then select your USB from the drop down menu, before hitting finish. This will bring up a screen for creating the image, select “Add” to chose the Image type, and select “Raw” as the other options are for specific Forensic software. A screen will appear were users can enter identifying information about the Image, followed by selecting the destination of the file, as seen in Figure 20. There are options to fragment the Image or compress it for less storage space, but that is not necessary in this demonstration, so set Fragmentation to 0.

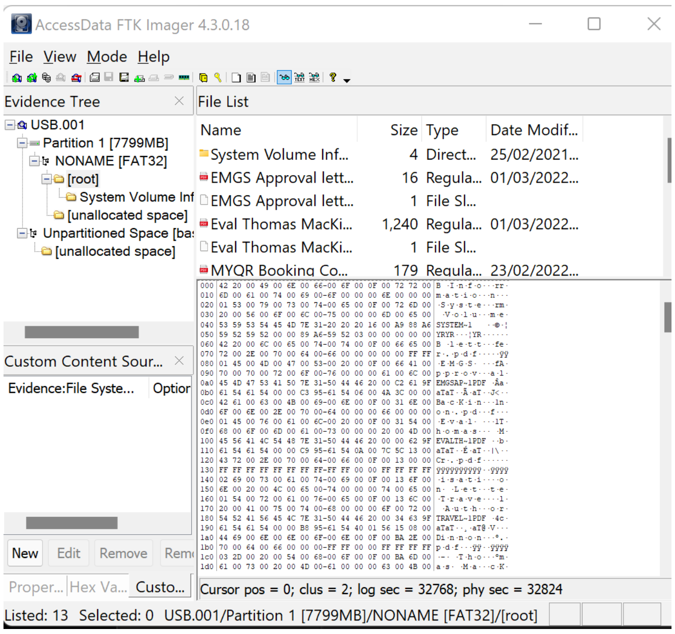


Case Number: DFCSTools1
Evidence Number: 1
Unique Description: Copy of suspect USB
Examiner: T. MacKinnon
Notes: Found hidden inside a Marble Statue of the Queen

Image Destination Folder: C:\Users\Tom\Documents\APU prereq\DF and CS tools
Image Filename (Excluding Extension): USB
Image Fragment Size (MB) For Raw, E01, and AFF formats: 0 = do not fragment: 1500
Compression (0=None, 1=Fastest, ..., 9=Smallest): 0
Use AD Encryption ☐

Figure 20: Options and Information screens for Creation of an Image in FTK Imager

Click "Finish" and then click "Start", which will result in a loading screen as FTK Imager makes the image. This may take a while, and FTK will verify the Image after completion, showing a Summary page. This Image can then be opened with FTK Imager to check if it worked, by using the "Add Evidence Item" and choosing the newly created image. This should result in something similar to Figure 21, and allow the Investigator free reign over the contents of the image.



```
USB.001
  Partition 1 [7799MB]
    NONAME [FAT32]
      [root]
        System Volume Inf
      [unallocated space]
  Unpartitioned Space [bas
    [unallocated space]
```

Name	Size	Type	Date Modif...
System Volume Inf...	4	Direct...	25/02/2021...
EMGS Approval lett...	16	Regula...	01/03/2022...
EMGS Approval lett...	1	File Sl...	
Eval Thomas MacKi...	1,240	Regula...	01/03/2022...
Eval Thomas MacKi...	1	File Sl...	
MYQR Booking Co...	179	Regula...	23/02/2022...

```
000 42 20 00 49 00 6E 00 66-00 6F 00 0F 00 72 72 00 B ·I·n·f·o···rr·
010 6D 00 61 00 74 00 69 00-6F 00 00 00 6E 00 00 00 m·a·t·i·o···n···
020 01 53 00 79 00 73 00 74-00 65 00 0F 00 72 6D 00 ·S·y·s·t·e···rm·
030 20 00 56 00 6F 00 6C 00-75 00 00 00 6D 00 65 00  ·V·o·l·u···m·e·
040 53 59 53 54 45 4D 7E 31-20 20 20 16 00 A9 88 A6 SYSTEM~1   ··©·¦
050 59 52 59 52 00 00 89 A6-59 52 03 00 00 00 00 00 YRYR···¦YR······
060 42 20 00 6C 00 65 00 74-00 74 00 0F 00 66 65 00 B ·l·e·t·t···fe·
070 72 00 2E 00 70 00 64 00-66 00 00 00 00 00 FF FF r·.·p·d·f·····ÿÿ
080 01 45 00 4D 00 47 00 53-00 20 00 0F 00 66 41 00 ·E·M·G·S· ···fA·
090 70 00 70 00 72 00 6F 00-76 00 00 00 61 00 6C 00 p·p·r·o·v···a·l·
0a0 45 4D 47 53 41 50 7E 31-50 44 46 20 00 C2 61 9F EMGSAP~1PDF ·Âa·
0b0 61 54 61 54 00 00 C3 95-61 54 06 00 4A 3C 00 00 aTaT··Ã·aT··J<··
0c0 42 61 00 63 00 4B 00 69-00 6E 00 0F 00 31 6E 00 Ba·c·K·i·n···1n·
0d0 6F 00 6E 00 2E 00 70 00-64 00 00 00 66 00 00 00 o·n·.·p·d···f···
0e0 01 45 00 76 00 61 00 6C-00 20 00 0F 00 31 54 00 ·E·v·a·l· ···1T·
0f0 68 00 6F 00 6D 00 61 00-73 00 00 00 20 00 4D 00 h·o·m·a·s··· ·M·
100 45 56 41 4C 54 48 7E 31-50 44 46 20 00 00 62 9F EVALTH~1PDF ··b·
110 61 54 61 54 00 00 C9 95-61 54 0A 00 7C 5C 13 00 aTaT··É·aT··|\··
120 43 72 00 2E 00 70 00 64-00 66 00 0F 00 13 00 00 Cr·.·p·d·f······
130 FF FF FF FF FF FF FF FF-FF FF 00 00 FF FF FF FF ÿÿÿÿÿÿÿÿÿÿ··ÿÿÿÿ
140 02 69 00 73 00 61 00 74-00 69 00 0F 00 13 6F 00 ·i·s·a·t·i····o·
150 6E 00 20 00 4C 00 65 00-74 00 00 00 74 00 65 00 n· ·L·e·t···t·e·
160 01 54 00 72 00 61 00 76-00 65 00 0F 00 13 6C 00 ·T·r·a·v·e····l·
170 20 00 41 00 75 00 74 00-68 00 00 00 6F 00 72 00  ·A·u·t·h···o·r·
180 54 52 41 56 45 4C 7E 31-50 44 46 20 00 34 63 9F TRAVEL~1PDF ·4c·
190 61 54 61 54 00 00 B8 95-61 54 40 01 56 15 08 00 aTaT··¸·aT@·V···
1a0 44 69 00 6E 00 6E 00 6F-00 6E 00 0F 00 BA 2E 00 Di·n·n·o·n···º.·
1b0 70 00 64 00 66 00 00 00-FF FF 00 00 FF FF FF FF p·d·f···ÿÿ··ÿÿÿÿ
1c0 03 2D 00 20 00 54 00 68-00 6F 00 0F 00 BA 6D 00 ·-· ·T·h·o···ºm·
1d0 61 00 73 00 20 00 4D 00-61 00 00 00 63 00 4B 00 a·s· ·M·a···c·K·
```

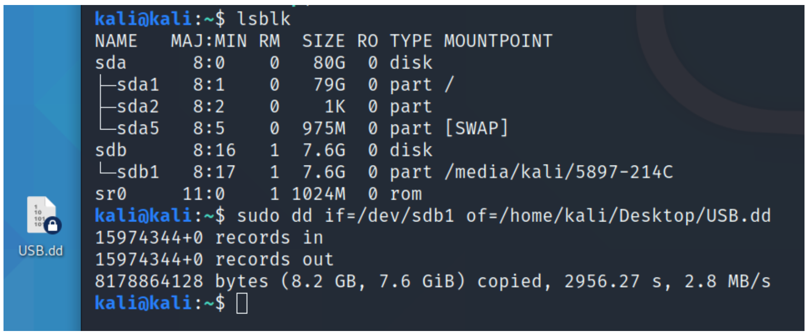
Cursor pos = 0; clus = 2; log sec = 32768; phy sec = 32824

Listed: 13 Selected: 0 USB.001/Partition 1 [7799MB]/NONAME [FAT32]/[root]

Figure 21: Successfully Investigating contents found on the Image of a USB, made with FTK Imager

3.2 Imaging in Linux

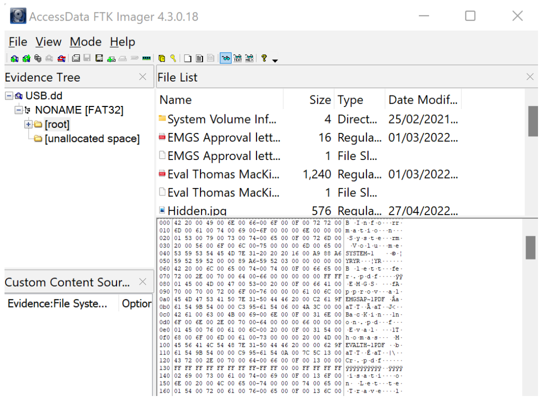
It is also worthwhile to learn how to Image storage devices in Linux, as it is the Operating System of choice for Digital Forensics Investigators and Cyber Security Specialists. There are many options for Imaging in Linux, DD tools is an Open source option that comes pre-installed on Kali Linux and is run in Command Line, so was selected for this demonstration (Parasram, 2020).



```
kali@kali:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0    80G  0 disk
├─sda1   8:1    0    79G  0 part /
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0   975M  0 part [SWAP]
sdb      8:16   1   7.6G  0 disk
└─sdb1   8:17   1   7.6G  0 part /media/kali/5897-214C
sr0     11:0    1 1024M  0 rom
kali@kali:~$ sudo dd if=/dev/sdb1 of=/home/kali/Desktop/USB.dd
15974344+0 records in
15974344+0 records out
8178864128 bytes (8.2 GB, 7.6 GiB) copied, 2956.27 s, 2.8 MB/s
kali@kali:~$ 
```

Figure 22: Successfully created an Image of a USB using DD in Kali Linux

To begin, insert or attach a USB to the Kali Linux operating system. Linux is less conventional when it comes to USB location, so to find out where it really is use the `lsblk` command, which will show devices connected to the PC. In Figure 22 the command reveals that the USB will be found at `/dev/sdb1` and is mounted to `/media/kali`. To use the dd command setting the "if flag" (Input file) as the USB location, and the "of flag" (Output Flag) as the destination of the image. In this example the command was `sudo dd if=/dev/sdb1 of=/home/kali/Desktop/USB.dd`, and produced the file after some time. This Image can be verified in FTK Imager, by adding it as evidence and browsing the contents, as seen in Figure 23.



Name	Size	Type	Date Modif...
System Volume Inf...	4	Direct...	25/02/2021...
EMGS Approval lett...	16	Regula...	01/03/2022...
EMGS Approval lett...	1	File Sl...	
Eval Thomas MacKi...	1,240	Regula...	01/03/2022...
Eval Thomas MacKi...	1	File Sl...	
Hidden.jpg	576	Regula...	27/04/2022...

```
000 42 20 00 49 00 6E 00 66-00 6F 00 0F 00 72 72 00 B ·I·n·f·o···rr·
010 6D 00 61 00 74 00 69 00-6F 00 00 00 6E 00 00 00 m·a·t·i·o···n···
020 01 53 00 79 00 73 00 74-00 65 00 0F 00 72 6D 00 ·S·y·s·t·e···rm·
030 20 00 56 00 6F 00 6C 00-75 00 00 00 6D 00 65 00  ·V·o·l·u···m·e·
040 53 59 53 54 45 4D 7E 31-20 20 20 16 00 A9 88 A6 SYSTEM~1   ··©·¦
050 59 52 59 52 00 00 89 A6-59 52 03 00 00 00 00 00 YRYR···¦YR·····
060 42 20 00 6C 00 65 00 74-00 74 00 0F 00 66 65 00 B ·l·e·t·t···fe·
070 72 00 2E 00 70 00 64 00-66 00 00 00 00 00 FF FF r·.·p·d·f·····ÿÿ
080 01 45 00 4D 00 47 00 53-00 20 00 0F 00 66 41 00 ·E·M·G·S· ···fA·
090 70 00 70 00 72 00 6F 00-76 00 00 00 61 00 6C 00 p·p·r·o·v···a·l·
0a0 45 4D 47 53 41 50 7E 31-50 44 46 20 00 C2 61 9F EMGSAP~1PDF ·Âa·
0b0 61 54 9B 54 00 00 C3 95-61 54 06 00 4A 3C 00 00 aT·T··Ã·aT··J<··
0c0 42 61 00 63 00 4B 00 69-00 6E 00 0F 00 31 6E 00 Ba·c·K·i·n···1n·
0d0 6F 00 6E 00 2E 00 70 00-64 00 00 00 66 00 00 00 o·n·.·p·d···f···
0e0 01 45 00 76 00 61 00 6C-00 20 00 0F 00 31 54 00 ·E·v·a·l· ···1T·
0f0 68 00 6F 00 6D 00 61 00-73 00 00 00 20 00 4D 00 h·o·m·a·s··· ·M·
100 45 56 41 4C 54 48 7E 31-50 44 46 20 00 00 62 9F EVALTH~1PDF ··b·
110 61 54 9B 54 00 00 C9 95-61 54 0A 00 7C 5C 13 00 aT·T··É·aT··|\··
120 43 72 00 2E 00 70 00 64-00 66 00 0F 00 13 00 00 Cr·.·p·d·f······
130 FF FF FF FF FF FF FF FF-FF FF 00 00 FF FF FF FF ÿÿÿÿÿÿÿÿÿÿ··ÿÿÿÿ
140 02 69 00 73 00 61 00 74-00 69 00 0F 00 13 6F 00 ·i·s·a·t·i····o·
150 6E 00 20 00 4C 00 65 00-74 00 00 00 74 00 65 00 n· ·L·e·t···t·e·
160 01 54 00 72 00 61 00 76-00 65 00 0F 00 13 6C 00 ·T·r·a·v·e····l·
```

Figure 23: FTK investigating the Image created with DD tools

3.3 Summary

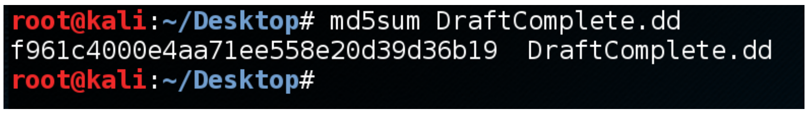
In summary, both tools used were able to perform the task of Imaging a USB in a forensically sound way. However, DD tools was far easier to use, as one command can create the Image, whilst FTK Imager required a lot of menus to get the task completed. DD tools was very fast compared to FTK Imager, taking around forty minutes to make an Image, whilst FTK took two and a half hours. FTK Imager is a lot easier for users with less technical skill, and didn't need the aid of online tutorials to use. However, in a real Digital Forensics Case DD tools is the obvious choice, only aided by Linux being the Industry standard operating system for Information Security.

4 Investigation Tools

An Investigation began after receiving a Memory Card from a Security Guard of a local Jewellery business. The guard had caught an employee trying to sneak out of the building with a Compact Memory Card concealed in a shoe. The guard suspects the employee was smuggling out building schematics and photos of products, which would be a great aid to any thieves planning to rob the business. The guard wishes to have evidence of this crime, and so has handed over the drive to be investigated.

4.1 Imaging and Hashing

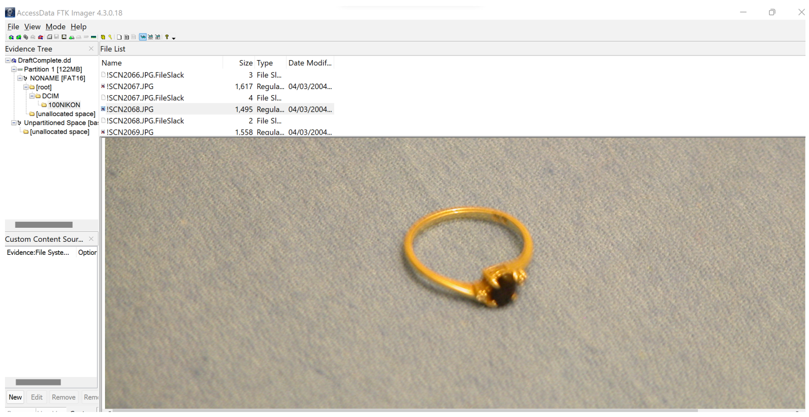
The first step of any investigation is to Image of the storage devices, luckily this step has already been performed. Still, a copy of the original Image will be used as to verify that no changes has been made. An MD5 hash will be take of the original file, which will be used to verify that during the Investigation no changes to the drive was made. To do this, simply use the command `md5sum filename.extension` as seen in Figure 24, revealing the hash signature of the original evidence.



```
root@kali:~/Desktop# md5sum DraftComplete.dd
f961c4000e4aa71ee558e20d39d36b19  DraftComplete.dd
root@kali:~/Desktop#
```

Figure 24: Checking the hash signature of DraftComplete.dd

4.2 Windows Investigation

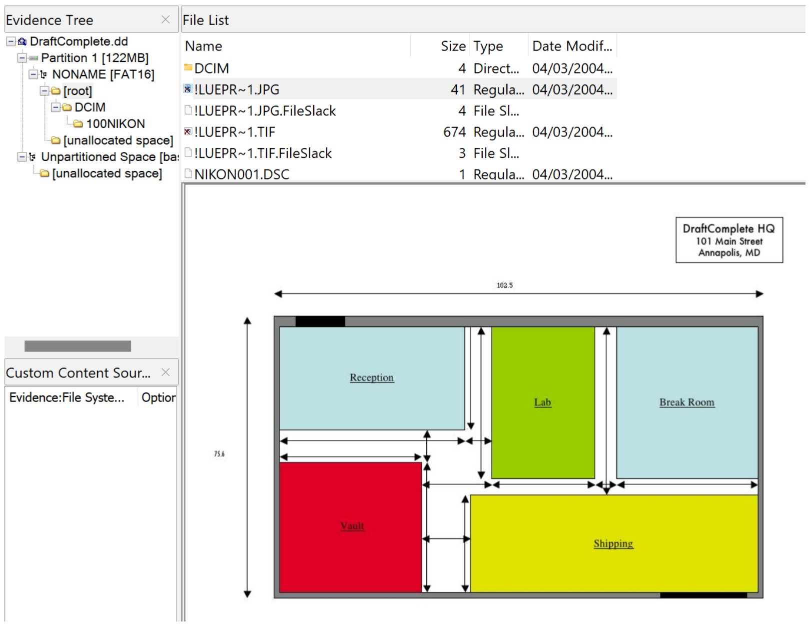


```
DraftComplete.dd
  Partition 1 [122MB]
    NONAME [FAT16]
      [root]
        DCIM
          100NIKON
        [unallocated space]
  Unpartitioned Space [ba
    [unallocated space]
```

Name	Size	Type	Date Modif...
!SCN2066.JPG.FileSlack	3	File Sl...	
!SCN2067.JPG	1,617	Regula...	04/03/2004...
!SCN2067.JPG.FileSlack	4	File Sl...	
!SCN2068.JPG	1,495	Regula...	04/03/2004...
!SCN2068.JPG.FileSlack	2	File Sl...	
!SCN2069.JPG	1,558	Regula...	04/03/2004...

Figure 25: FTK Imager investigating "DraftComplete.dd", revealing pictures of unreleased products

For the Windows portion of this Investigation FTK Imager will be used again to investigate the the Draft Complete Image. The Image was loaded into FTK by using the "Add Evidence Item" and then selecting "Image File" and choosing the "DraftComplete.dd" from the file manager. Searching through the content reveals a folder named "100NIKON", which has the name of a popular brand of camera. Looking into the folder reveals photos of unreleased Draft Complete Products. Figure 25 shows an image retrieved of a ring, there are several other images of products, including one saved as TIFF file which can't be viewed in FTK until it is extracted. All evidence uncovered can be found in the Appendix.


Evidence Tree
DraftComplete.dd
Partition 1 [122MB]
NONAME [FAT16]
[root]
DCIM
100NIKON
[unallocated space]
Unpartitioned Space [ba
[unallocated space]

File List

Name	Size	Type	Date Modif...
DCIM	4	Direct...	04/03/2004...
!LUEPR~1.JPG	41	Regula...	04/03/2004...
!LUEPR~1.JPG.FileSlack	4	File Sl...	
!LUEPR~1.TIF	674	Regula...	04/03/2004...
!LUEPR~1.TIF.FileSlack	3	File Sl...	
NIKON001.DSC	1	Regula...	04/03/2004...

Custom Content Sour...
Evidence:File Syste... Option

Figure 26: FTK Imager investigating "DraftComplete.dd", revealing building blueprints of Draft Complete's HQ

Another notable file found on the drive can be seen in Figure 26, being the building blueprints for Draft Complete's HQ. This is particularly worrying, as these files would be highly sought after by any criminal planning to rob the building. This concluded the Windows investigation, the hash signatures were checked and were the same, meaning all evidence retrieved was not accidentally tampered with by the investigator.

4.3 Linux Investigation

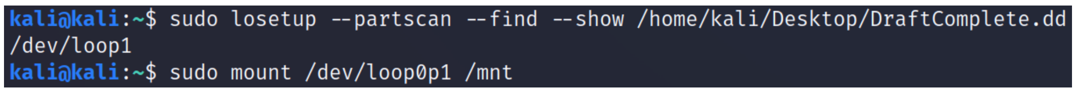

```
kali@kali:~$ sudo losetup --partscan --find --show /home/kali/Desktop/DraftComplete.dd
/dev/loop1
kali@kali:~$ sudo mount /dev/loop0p1 /mnt
```

Figure 27: Mounting Command to mount "DraftComplete.dd" to Kali Linux

To begin the Linux investigation the drive was mounted to the Operating System so it could be browsed through. At first the `losetup` command with the "find" flag was used to find and set up "DraftComplete.dd" as a loop device. Once a loop devices is associated with the drive it can be mounted with the mount command, as seen in Figure 27.

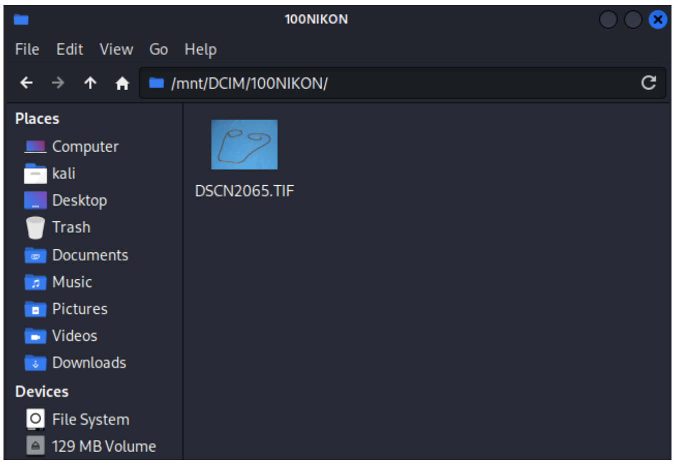
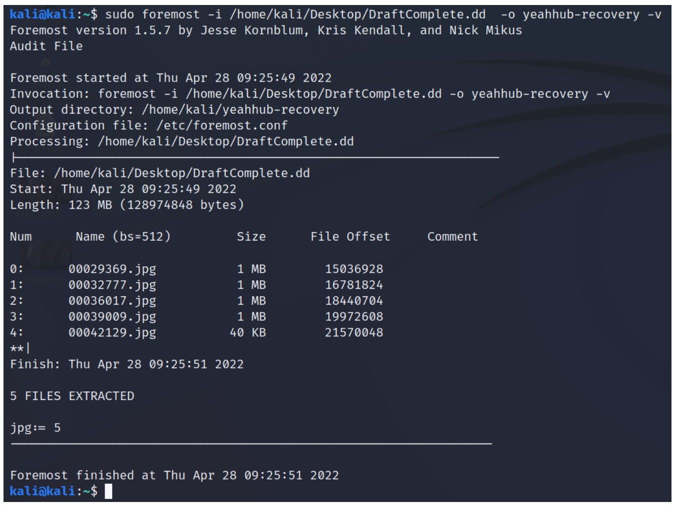


Figure 28: Successfully mounted "DraftComplete.dd"

To verify mounting has worked browse to the file location of the drive, in this being in the `mnt` folder of the root file system. Inside reveals little of the known content for "DraftComplete.dd" from FTK Imager, with only one product image present. This confirms that the suspect employee had deleted files off the usb.

To recover the file Foremost was used with the Yeahhub recovery option, which can recover files using a process known as file carving. The full command can be seen in Figure 29, the tool works quickly and output to `/home/kali/yeahhub-recovery`, however, the files permissions do not allow viewing of contents. To fix this the output file permissions were changed using ChMod using the command `sudo chmod 777 /home/kali/yeahhub-recovery`. This will give full access to the contents.



```
kali@kali:~$ sudo foremost -i /home/kali/Desktop/DraftComplete.dd  -o yeahhub-recovery -v
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Thu Apr 28 09:25:49 2022
Invocation: foremost -i /home/kali/Desktop/DraftComplete.dd -o yeahhub-recovery -v
Output directory: /home/kali/yeahhub-recovery
Configuration file: /etc/foremost.conf
Processing: /home/kali/Desktop/DraftComplete.dd
|------------------------------------------------------------------
File: /home/kali/Desktop/DraftComplete.dd
Start: Thu Apr 28 09:25:49 2022
Length: 123 MB (128974848 bytes)

Num      Name (bs=512)         Size      File Offset     Comment

0:      00029369.jpg           1 MB        15036928
1:      00032777.jpg           1 MB        16781824
2:      00036017.jpg           1 MB        18440704
3:      00039009.jpg           1 MB        19972608
4:      00042129.jpg          40 KB        21570048
**|
Finish: Thu Apr 28 09:25:51 2022

5 FILES EXTRACTED

jpg:= 5
------------------------------------------------------------------

Foremost finished at Thu Apr 28 09:25:51 2022
kali@kali:~$
```

Figure 29: Foremost command with Yeahhub recovery option

All files that were deleted are now recovered as seen in Figure 30, thus concluding this investigation. The hash signatures were checked revealing no changes had been made to DD file, meaning the integrity of the evidence had been maintained throughout this investigation.

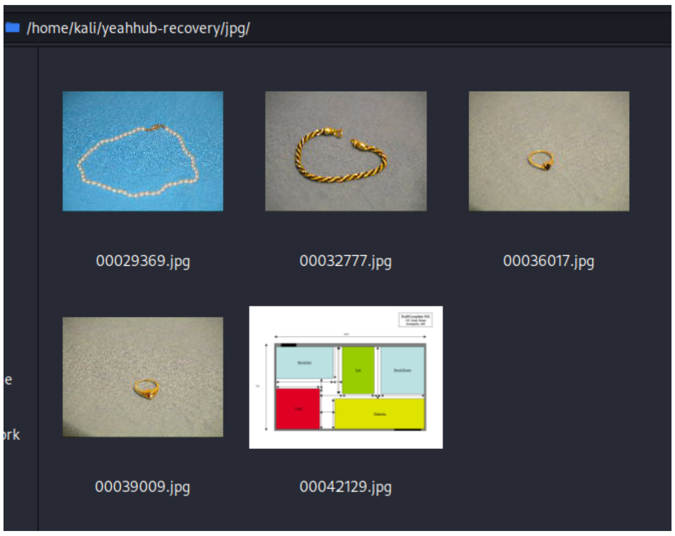


Figure 30: Foremost recovering deleted files from “DraftComplete.dd”

4.4 Summary

From this investigation it is clear that the suspect employee had in their possession a Compact Flash memory card that contained confident files, such as unreleased product pictures and HQ building blueprints. The contents had been deleted in an attempt to hide evidence of misconduct, but could easily be recovered using file carving tools. This suggests that the suspect planned to recover files after smuggling them out of the building, proving the suspicions of the Security Guard.

5 Conclusion

In Conclusion, this report successfully covers many tools and techniques used by professionals of Cyber Security and Digital Forensics. Many aspects of Information Security was covered, from hexediting to steganography, providing an informative rundown of tools. The use of a Case Study provides a practical use for tools covered and proves the true usefulness they can provide in the persuit of convicting criminals and providing a safer cyberspace for us all.

6 References

Al-Matari, O.M., Elhennawy, S., Helal, I.M.A., Mazen, S.A. 2018. Cybersecurity Tools for IS Auditing. *2018 Sixth International Conference on Enterprise Systems (ES)*. pp. 217-223.

CloudNine. N.D. How to Create an Image Using FTK Imager - eDiscovery Best Practices. [online] Cloud Nine. Available at:
https://cloudnine.com/ediscoverydaily/electronic-discovery/
how-to-create-an-image-using-ftk-imager-ediscovery-best-practices/ [Accessed 28/04/2022]

FBI. 2021. FBI Releases the Internet Crime Complaint Center 2020 Internet Crime Report, Including COVID-19 Scam Statistics. [online] FBI Government Site. Available at:
https://www.fbi.gov/news/pressrel/press-releases/
fbi-releases-the-internet-crime-complaint-center-2020-internet-crime-report-including-covid-19-scam-statistics
[Accessed 13 April 2022]

FBI. 2020. Internet Crime Report 2020. Pennsylvania, United States of America. Available at:
https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf [Accessed 13 April 2022]

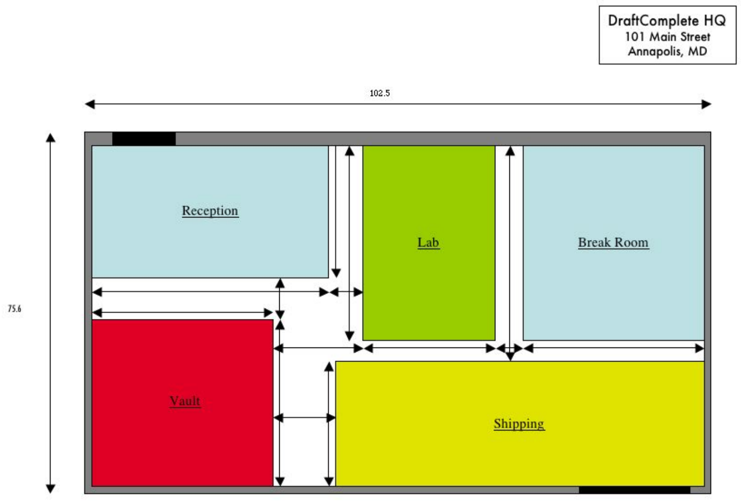
GCHQ. N.D. Cyber Chef. [online] GCHQ Github. Available at:
https://gchq.github.io/CyberChef/ [Accessed 28/04/2022]

ODS. 2017. What does metadata disclose in photos?. [online] Open Data Security. Available at:
https://opendatasecurity.co.uk/what-does-metadata-disclose-in-photos/ [Accessed 17/04/2022]

Parasram, S.V.N. 2020. Evidence Acquisition and Preservation with dc3dd and Guymager. *Digital Forensics with Kali Linux Second Edition.* Birmingham, United Kingdom, Packt Publishing.

7 Appendix

7.1 Draft Complete



DraftComplete HQ
101 Main Street
Annapolis, MD
102.5
75.6
Reception
Lab
Break Room
Vault
Shipping