Network Forensics Investigation

Lukas Smith
Year 4 Ethical Hacking
CMP416 Advanced Digital Forensics
2022/23

Contents

Introduction

Scenario

This report details the findings of a Network Forensics Investigation undertaken on behalf of a National Security Agency regarding an International Sporting Corruption case. Three network captures have been exfiltrated from parties of interest and the purpose of the investigation was to recover the requested information from each capture. The report will address the methodology and investigative techniques undertaken to prove the investigation was forensically sound. Additionally this report will detail the tools used for the investigation and what network data was valuable for the investigation. Finally, a critical evaluation of the investigation will be discussed, with a reflection on how a malicious actor could have impeded the investigation.

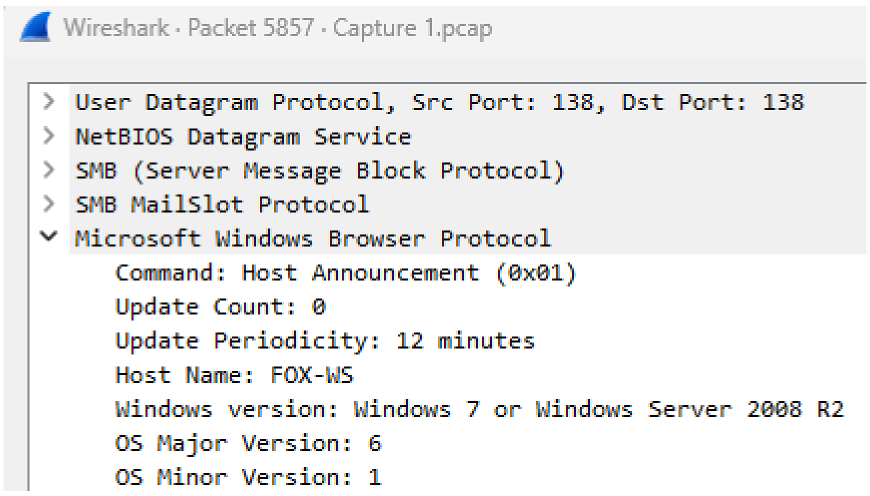
Investigative Tools

Tool Name	**Description**
binwalk	A tool that is pre-installed on Kali Linux, it can search binary images for embedded files and executable code.
CyberChef	A web application used for encryption, encoding, compression and data analysis. It was used to decode some encoded files found in Capture 1.
Kali Linux	Kali Linux is a Linux distribution with the tools for Digital Fornesics and Penetration Testing pre-installed.
md5sum	A command pre-installed on Kali Linux used to ensure the files obtained had not been tampered with during the investigation.
SilentEye	A cross-platform steganography tool used to hide text or files within images and sounds. This was used to find an encoded hidden message in Capture 2.
tshark	A command-line based network protocol analyser tool which is pre-installed on Kali Linux. This tool provides more technical commands than wireshark, meaning it can be used for retrieving more specific evidence.
Visual Studio Code	The IDE used to fix the 'broken.py' file found in Capture 1, and to create the script used to identify the date of the meeting in Capture 3.
Wireshark	An open-source packet analyser used in this case to extract potential evidence. Evidence can be extract either by directly exporting stored bytes or by using the 'Export Objects' command.

Capture 1.pcap

Initial Packet Analysis

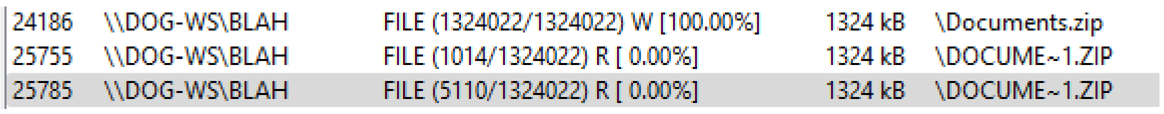
The investigator was made aware that this capture contained files downloaded by the suspect, so the first step was to identify protocols in the capture that could be used for file download. Using wireshark to analyse the pcap file, the protocol which stood out was SMB. Server Message Block or SMB is a communication protocol which provides shared access to files across points on a network. The first SMB packet is a Host Announcement, which includes the hostname 'FOX-WS', Wireshark identifies the OS as Windows 7 or Windows Server 2008 R2.



```
Wireshark · Packet 5857 · Capture 1.pcap

> User Datagram Protocol, Src Port: 138, Dst Port: 138
> NetBIOS Datagram Service
> SMB (Server Message Block Protocol)
> SMB MailSlot Protocol
v Microsoft Windows Browser Protocol
    Command: Host Announcement (0x01)
    Update Count: 0
    Update Periodicity: 12 minutes
    Host Name: FOX-WS
    Windows version: Windows 7 or Windows Server 2008 R2
    OS Major Version: 6
    OS Minor Version: 1
```

Figure 1 - The Host connected to the SMB with the Windows version

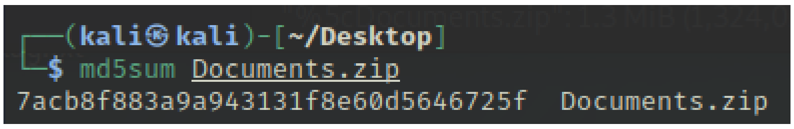
Wireshark makes it easy to identify and capture files downloaded within the capture with the `Export Objects -> SMB…` command. There were 9 objects listed from the SMB server identified as "DOG-WS", the dll file 'srvsvc' and the ini files can be ignored as the dll is a service file and the ini files are folder preference files. Two of the three remaining files are zip files and appear to be incomplete downloads of the final zip 'Documents.zip', as they have the same final number of bytes and come from the same directory.



24186	\\DOG-WS\BLAH	FILE (1324022/1324022) W [100.00%]	1324 kB	\Documents.zip
25755	\\DOG-WS\BLAH	FILE (1014/1324022) R [0.00%]	1324 kB	\DOCUME~1.ZIP
25785	\\DOG-WS\BLAH	FILE (5110/1324022) R [0.00%]	1324 kB	\DOCUME~1.ZIP

Figure 2 - Documents.zip and the incomplete file downloads

'Documents.zip' was saved and immediately a hash was generated for the file to ensure it remains unmodified during the investigation. (Kumar, et al., 2012)

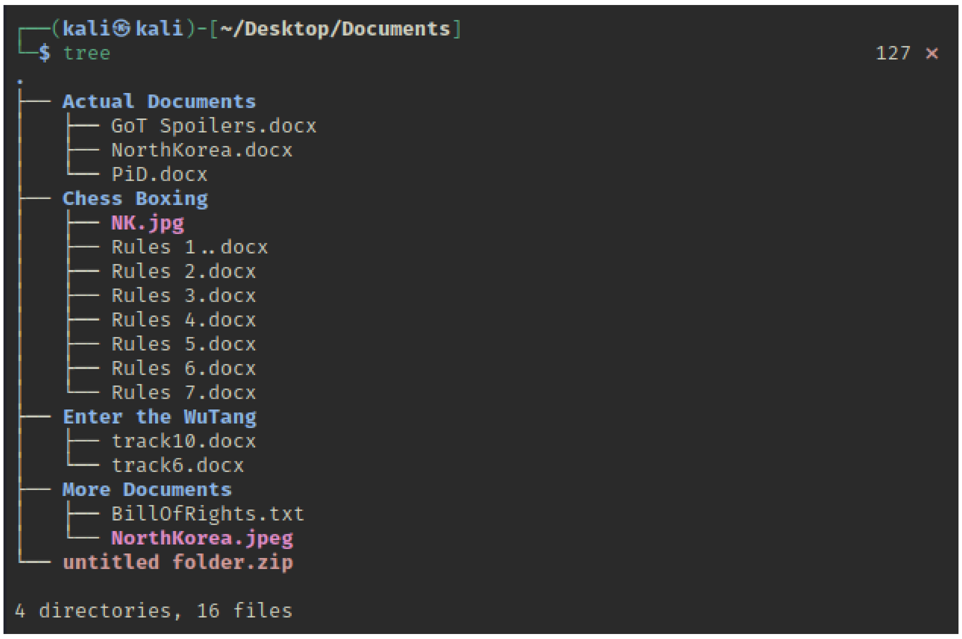


```
┌──(kali㉿kali)-[~/Desktop]
└─$ md5sum Documents.zip
7acb8f883a9a943131f8e60d5646725f  Documents.zip
```

Figure 3 - Documents.zip MD5 Hash

Documents.zip Investigation

Documents.zip had several directories inside of it and another zipped file called 'untitled folder.zip'



```
┌──(kali㉿kali)-[~/Desktop/Documents]
└─$ tree                                                    127 ×
.
├── Actual Documents
│   ├── GoT Spoilers.docx
│   ├── NorthKorea.docx
│   └── PiD.docx
├── Chess Boxing
│   ├── NK.jpg
│   ├── Rules 1..docx
│   ├── Rules 2.docx
│   ├── Rules 3.docx
│   ├── Rules 4.docx
│   ├── Rules 5.docx
│   ├── Rules 6.docx
│   └── Rules 7.docx
├── Enter the WuTang
│   ├── track10.docx
│   └── track6.docx
├── More Documents
│   ├── BillOfRights.txt
│   └── NorthKorea.jpeg
└── untitled folder.zip

4 directories, 16 files
```

Figure 4 - File Tree Structure of 'Documents.zip'

Actual Documents

This directory had three word documents inside it which were encoded in Base64. The files were decoded using CyberChef, the decoded formats can be found at **A1 – Capture 1/Actual Documents/**. According to the info tab on word, all the files were 'Authored by Eric' and 'Last Modified by Bryan Schmidt'. After decoding the text contained in these files, it was found that they are not relevant to the National Security Agencies investigation but have been included in the Appendixes.

Chess Boxing

All the word documents (Rules 1. to Rules 7) are the rules of chess boxing encoded in Base64. This has been decoded and put into one word document called 'Chess Rules.docx' and can be found at

A2 – Capture 1/Chess Boxing/.

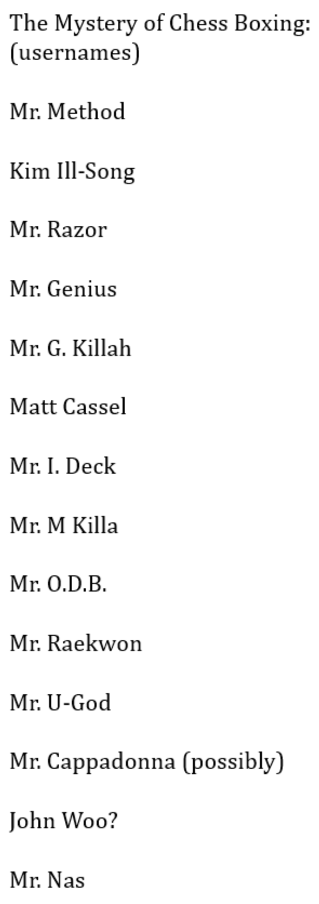
NK.jpg is a JPEG of the North Korea flag.

Enter the WuTang

The two word documents in this directory have their 'Authors listed as Bryan Schmidt' and 'Last Modified by Bryan Schmidt' and have been encoded in Base64. These have been decoded using CyberChef and can be found in **A3 – Capture1/Enter the WuTang**.

The first file 'track10.docx' is song lyrics.

The second file 'track6.docx' is a list of usernames. It can be assumed that these usernames are the suspected list of potential names/aliases of actors involved in the corruption case.



The Mystery of Chess Boxing:
(usernames)

Mr. Method

Kim Ill-Song

Mr. Razor

Mr. Genius

Mr. G. Killah

Matt Cassel

Mr. I. Deck

Mr. M Killa

Mr. O.D.B.

Mr. Raekwon

Mr. U-God

Mr. Cappadonna (possibly)

John Woo?

Mr. Nas

Figure 5 - track6.docx decoded, with suspect usernames

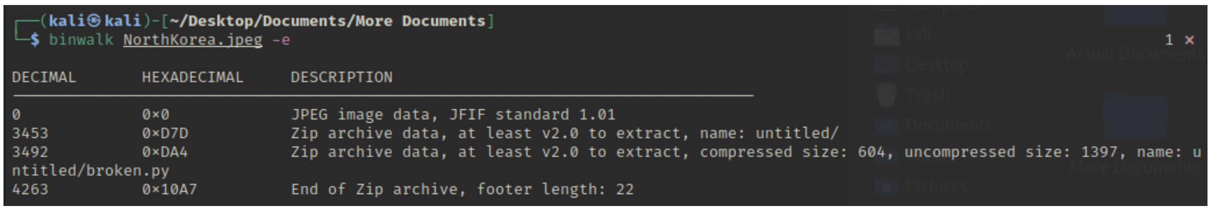
More Documents

'BillOfRights.txt' appears to be a text file containing the US Constitution, also known as the Bill of Rights.

'NorthKorea.jpg' appears to be a JPEG of the North Korea flag.

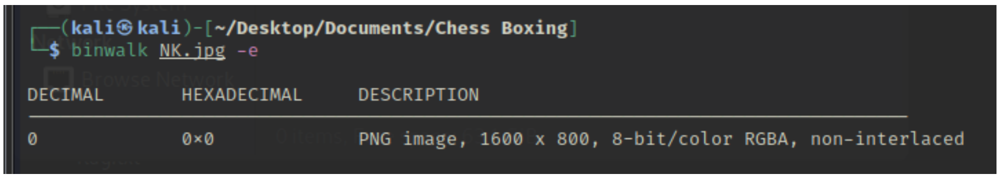
untitled folder.zip

This zip file contains several untitled folders inside of themselves, with the final folder being called 'Silent Eye'. On researching 'Silent Eye' it was discovered to be steganography tool for hiding data within images and files, including JPEGs. This suggested to the investigator that Steganography may be being used in some of the found images. The command 'binwalk -e' was used on both images found, 'NorthKorea.jpeg' and 'NK.jpg' to identify and extract any possible data hidden within them.



```
┌──(kali㉿kali)-[~/Desktop/Documents/More Documents]
└─$ binwalk NorthKorea.jpeg -e                                                  1 ×

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
3453          0xD7D           Zip archive data, at least v2.0 to extract, name: untitled/
3492          0xDA4           Zip archive data, at least v2.0 to extract, compressed size: 604, uncompressed size: 1397, name: u
ntitled/broken.py
4263          0x10A7          End of Zip archive, footer length: 22
```

Figure 6 - binwalk on 'NorthKorea.jpeg'



```
┌──(kali㉿kali)-[~/Desktop/Documents/Chess Boxing]
└─$ binwalk NK.jpg -e

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1600 x 800, 8-bit/color RGBA, non-interlaced
```

Figure 7 - binwalk on 'NK.jpg'

Nothing was found within ‘NK.jpg’ but binwalk did extract a ZIP file and a python file from ‘NorthKorea.jpeg’. The ZIP file contained an exact copy of the python file called ‘broken.py,’ which appears to be a text encoder with syntax errors. It can be found within **A4 – Capture1/More Documents/.**

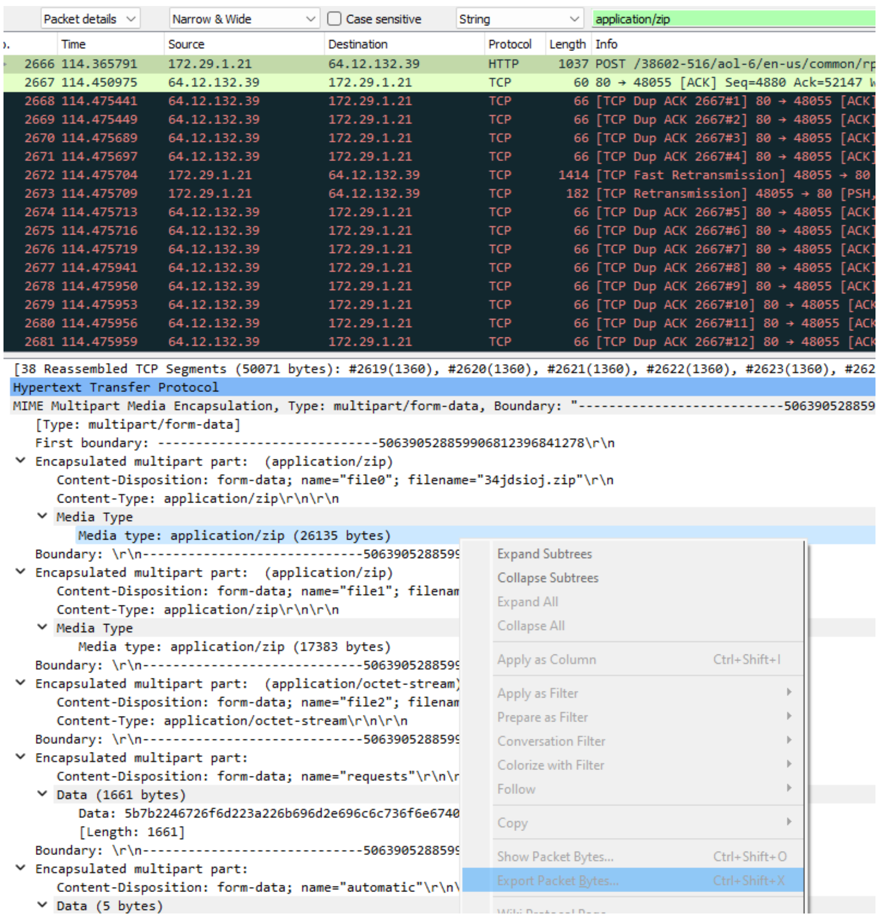
Critical Evaluation

Originally the investigator attempted to recover messages from the images found using the program ‘SilentEye’ however these returned errors as they were not the correct format for the program. Instead, the investigator used binwalk as it is a well known steganography recovery tool.

Capture 2.pcap

Initial Packet Analysis

The investigator was made aware that traffic captured in the files contains files transferred between a suspected corrupt official and a foreign national. It also hinted that one of the traffic protocols used was FTP. By using Wiresharks ‘Export Objects -> FTP-DATA’ command, two ZIP files were recovered. Upon inspecting these files it was found to be a series of invalid ‘.jpg’s. Another part of the brief suggests that anti-forensics practises were used and that an Edward Snowden quote would help deciphering. Using the word ‘conscience,’ the quote was found and it was determined that there were words missing. To find these missing words, the pcap file was combed again and three additional zip files were recovered. These zip can be found at **B1 – Capture 2/Zips/**.

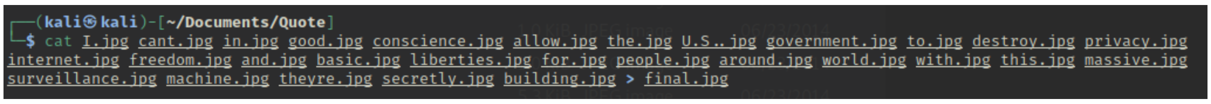


```
 .   Time        Source         Destination    Protocol Length Info
2666 114.365791  172.29.1.21    64.12.132.39   HTTP     1037 POST /38602-516/aol-6/en-us/common/rp
2667 114.450975  64.12.132.39   172.29.1.21    TCP        60 80 → 48055 [ACK] Seq=4880 Ack=52147 W
2668 114.475441  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#1] 80 → 48055 [ACK]
2669 114.475449  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#2] 80 → 48055 [ACK]
2670 114.475689  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#3] 80 → 48055 [ACK]
2671 114.475697  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#4] 80 → 48055 [ACK]
2672 114.475704  172.29.1.21    64.12.132.39   TCP      1414 [TCP Fast Retransmission] 48055 → 80
2673 114.475709  172.29.1.21    64.12.132.39   TCP       182 [TCP Retransmission] 48055 → 80 [PSH,
2674 114.475713  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#5] 80 → 48055 [ACK]
2675 114.475716  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#6] 80 → 48055 [ACK]
2676 114.475719  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#7] 80 → 48055 [ACK]
2677 114.475941  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#8] 80 → 48055 [ACK]
2678 114.475950  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#9] 80 → 48055 [ACK]
2679 114.475953  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#10] 80 → 48055 [ACK
2680 114.475956  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#11] 80 → 48055 [ACK
2681 114.475959  64.12.132.39   172.29.1.21    TCP        66 [TCP Dup ACK 2667#12] 80 → 48055 [ACK

[38 Reassembled TCP Segments (50071 bytes): #2619(1360), #2620(1360), #2621(1360), #2622(1360), #2623(1360), #262
Hypertext Transfer Protocol
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "----------------------------506390528859
   [Type: multipart/form-data]
   First boundary: ----------------------------506390528859906812396841278\r\n
 ˅ Encapsulated multipart part:  (application/zip)
     Content-Disposition: form-data; name="file0"; filename="34jdsioj.zip"\r\n
     Content-Type: application/zip\r\n\r\n
   ˅ Media Type
       Media type: application/zip (26135 bytes)
   Boundary: \r\n----------------------------506390528859
 ˅ Encapsulated multipart part:  (application/zip)
     Content-Disposition: form-data; name="file1"; filenam
     Content-Type: application/zip\r\n\r\n
   ˅ Media Type
       Media type: application/zip (17383 bytes)
   Boundary: \r\n----------------------------506390528859
 ˅ Encapsulated multipart part:  (application/octet-stream)
     Content-Disposition: form-data; name="file2"; filenam
     Content-Type: application/octet-stream\r\n\r\n
   Boundary: \r\n----------------------------506390528859
 ˅ Encapsulated multipart part:
     Content-Disposition: form-data; name="requests"\r\n\r
   ˅ Data (1661 bytes)
       Data: 5b7b2246726f6d223a226b696d2e696c6c736f6e6740
       [Length: 1661]
   Boundary: \r\n----------------------------506390528859
 ˅ Encapsulated multipart part:
     Content-Disposition: form-data; name="automatic"\r\n\
   ˅ Data (5 bytes)
```

Figure 8 - Recovered ZIP file

Steganography

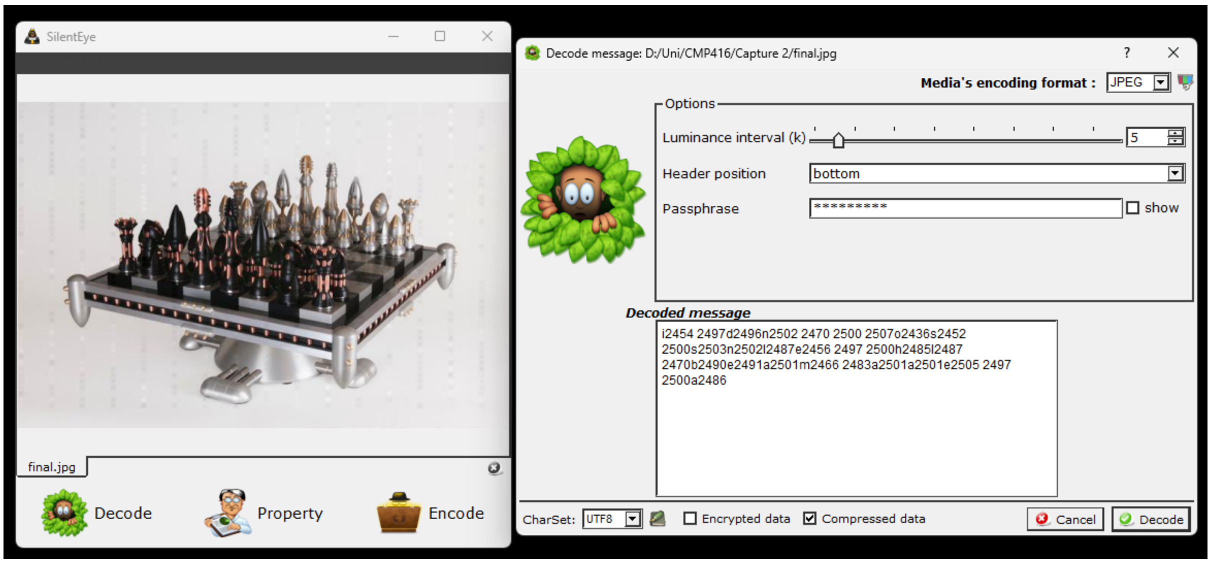
Many files start with 'magic bytes' that identify the filetype, for JPEGS the magic bytes are 'FF D8 FF'. Three of the images found had these file bytes, the rest did not. This indicates that these files have been manipulated by splitting the raw hex bytes at certain points. One of these images was recovered by matching the filenames to the Edward Snowden quote. The second 'the' was missed out in the quote.



```
┌──(kali㉿kali)-[~/Documents/Quote]
└─$ cat I.jpg cant.jpg in.jpg good.jpg conscience.jpg allow.jpg the.jpg U.S..jpg government.jpg to.jpg destroy.jpg privacy.jpg internet.jpg freedom.jpg and.jpg basic.jpg liberties.jpg for.jpg people.jpg around.jpg world.jpg with.jpg this.jpg massive.jpg surveillance.jpg machine.jpg theyre.jpg secretly.jpg building.jpg > final.jpg
```

Figure 9 - Recovering the file

Then, using the software that was found in the first capture 'SilentEye' a message was found within the image.



Decode message: D:/Uni/CMP416/Capture 2/final.jpg

Media's encoding format : JPEG

Options

Luminance interval (k) 5

Header position bottom

Passphrase ********* ☐ show

Decoded message

```
i2454 2497d2496n2502 2470 2500 2507o2436s2452
2500s2503n2502l2487e2456 2497 2500h2485l2487
2470b2490e2491a2501m2466 2483a2501a2501e2505 2497
2500a2486
```

CharSet: UTF8 ☐ Encrypted data ☑ Compressed data

Figure 10 - SilentEye Secret Message

This message appears to be a cipher- to decode it the Python program found in Capture 1 was used. This python file had syntax errors so was fixed before using. The program required a phrase for encoding and decoding. A few different phrases were attempted but the program mentioned the word 'bill' multiple times and the filename 'BillOfRights' that was found in the same folder as the python file, was found to be the correct phrase. The fixed python script can be found at **A5 – Capture1/fixed.py**.

When running the decoded message through the program with the passphrase 'BillOfRights,' the phrase 'DontTry2BruteForceThisPassword' was found.

Critical Evaluation

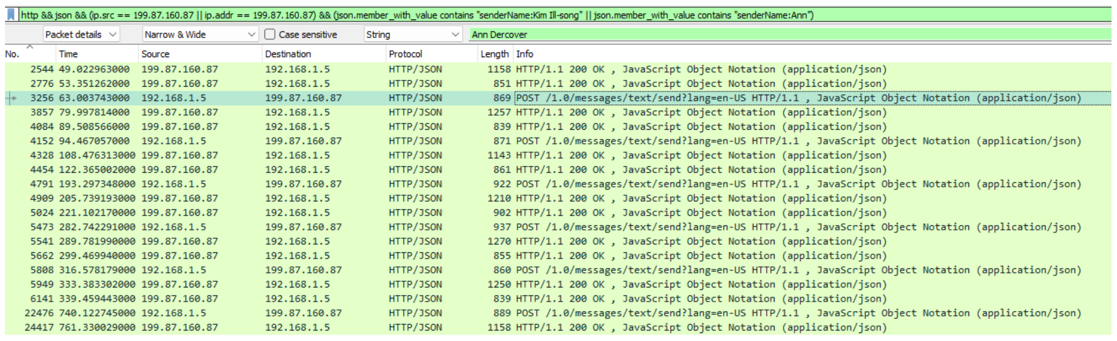
One of the main challenges encountered whilst attempting to recover the image from the fragmented image bytes, was the Edward Snowden quote. The quote features two 'the's but there was only one 'the.jpg' file recovered. Initially, using the same 'the.jpg' file was attempted and this produced a strange looking image that the investigator assumed to be incorrect. This was fixed by removing the second 'the.jpg' file.

Capture 3.pcap

Initial Packet Analysis

This capture contained a conversation between one 'Ann Dercover' and 'Ill Song'. The conversation was found over several json files in the HTTP protocol. The IP address of 'Ill Song' was identified and used to create a filter to only display the relevant traffic.

```
'http && json && (ip.src == 199.87.160.87 || ip.addr == 199.87.160.87) &&
(json.member_with_value contains "senderName:Kim Ill-song" ||
json.member_with_value contains "senderName:Ann")'
```



```
http && json && (ip.src == 199.87.160.87 || ip.addr == 199.87.160.87) && (json.member_with_value contains "senderName:Kim Ill-song" || json.member_with_value contains "senderName:Ann")
Packet details    Narrow & Wide    Case sensitive    String    Ann Dercover

No.   Time          Source          Destination     Protocol    Length Info
 2544 49.022963000  199.87.160.87   192.168.1.5     HTTP/JSON    1158 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 2776 53.351262000  199.87.160.87   192.168.1.5     HTTP/JSON     851 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 3256 63.003743000  192.168.1.5     199.87.160.87   HTTP/JSON     869 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
 3857 79.997814000  199.87.160.87   192.168.1.5     HTTP/JSON    1257 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 4084 89.508566000  199.87.160.87   192.168.1.5     HTTP/JSON     839 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 4152 94.467057000  192.168.1.5     199.87.160.87   HTTP/JSON     871 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
 4328 108.476313000 199.87.160.87   192.168.1.5     HTTP/JSON    1143 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 4454 122.365002000 199.87.160.87   192.168.1.5     HTTP/JSON     861 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 4791 193.297348000 192.168.1.5     199.87.160.87   HTTP/JSON     922 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
 4909 205.739193000 199.87.160.87   192.168.1.5     HTTP/JSON    1210 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 5024 221.102170000 199.87.160.87   192.168.1.5     HTTP/JSON     902 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 5473 282.742291000 192.168.1.5     199.87.160.87   HTTP/JSON     937 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
 5541 289.781990000 199.87.160.87   192.168.1.5     HTTP/JSON    1270 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 5662 299.469940000 199.87.160.87   192.168.1.5     HTTP/JSON     855 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 5808 316.578179000 192.168.1.5     199.87.160.87   HTTP/JSON     860 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
 5949 333.383302000 199.87.160.87   192.168.1.5     HTTP/JSON    1250 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 6141 339.459443000 199.87.160.87   192.168.1.5     HTTP/JSON     839 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
22476 740.122745000 192.168.1.5     199.87.160.87   HTTP/JSON     889 POST /1.0/messages/text/send?lang=en-US HTTP/1.1 , JavaScript Object Notation (application/json)
24417 761.330029000 199.87.160.87   192.168.1.5     HTTP/JSON    1158 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
```

Figure 11 - Filtered Traffic

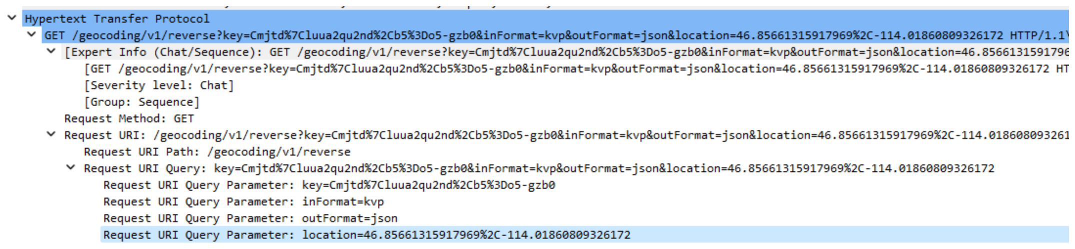
The JSON files were extracted from these packets to retrieve the conversation. The full list of JSON files can be found at **C1 – Capture 3/Messages/**. The messages were as follows:

Author	Content
Kim Ill-song	Good afternoon, Ann.
Ann	who is this?
Kim Ill-song	Castling.
Ann	where are you?
Kim Ill-song	I know I can't tell you that.
Ann	Do you know that there are people investigating Kim Ill-Song?
Kim Ill-song	Of course. However, they will never know it is me behind the bribes.
Ann	still we should be careful. Pay attention. I want to meet in September at 5PM.
Kim Ill-song	At our old meetup spot?
Ann	yes
Kim Ill-song	What day?
Ann	I told you to pay attention.

From these messages it can be determined that ‘Kim Ill-song’, a username found in the list of usernames in Capture 1, has another identity ‘Castling’. In addition, when ‘Castling’ asks what day they are going to meet up, Ann replies ‘pay attention’. This suggests there is something else to investigate.

Further Analysis

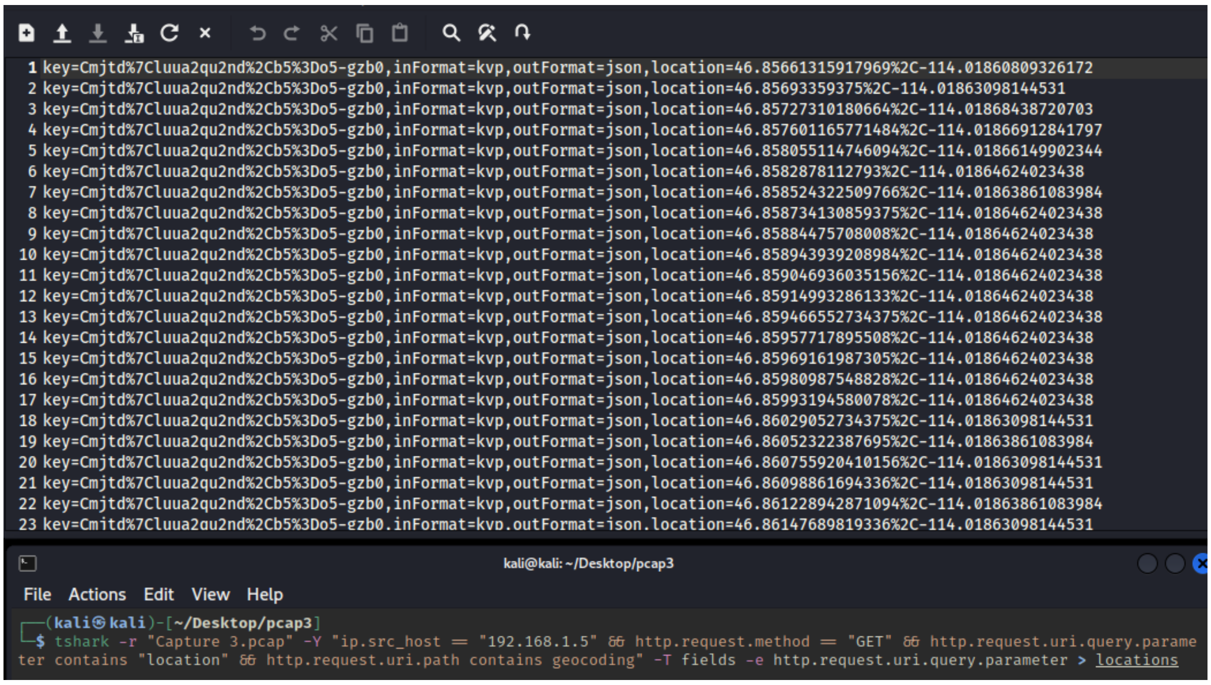
On further investigating the PCAP, a significant amount of GET requests to a map website were found. In the HTTP part of the ‘geocoding’ packets, the location was stored.



```
Hypertext Transfer Protocol
  GET /geocoding/v1/reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85661315917969%2C-114.01860809326172 HTTP/1.1
    [Expert Info (Chat/Sequence): GET /geocoding/v1/reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.856613159179
      [GET /geocoding/v1/reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85661315917969%2C-114.01860809326172 HT
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI: /geocoding/v1/reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85661315917969%2C-114.01860809326
      Request URI Path: /geocoding/v1/reverse
      Request URI Query: key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85661315917969%2C-114.01860809326172
        Request URI Query Parameter: key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0
        Request URI Query Parameter: inFormat=kvp
        Request URI Query Parameter: outFormat=json
        Request URI Query Parameter: location=46.85661315917969%2C-114.01860809326172
```

Figure 12 - Location in Latitude and Longitude

Using tshark, a command line network analyser, the ‘Request URI Query Parameter’s header for each packet matching the word ‘geocoding’ were dumped to a text file.



```
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85661315917969%2C-114.01860809326172
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85693359375%2C-114.01863098144531
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85727310180664%2C-114.01868438720703
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.857601165771484%2C-114.01866912841797
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.858055114746094%2C-114.01866149902344
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.8582878112793%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.858524322509766%2C-114.01863861083984
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.858734130859375%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85884475708008%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.858943939208984%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.859046936035156%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85914993286133%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.859466552734375%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85957717895508%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85969161987305%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85980987548828%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.85993194580078%2C-114.01864624023438
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.86029052734375%2C-114.01863098144531
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.86052322387695%2C-114.01863861083984
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.860755920410156%2C-114.01863098144531
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.86098861694336%2C-114.01863098144531
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.861228942871094%2C-114.01863861083984
key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0,inFormat=kvp,outFormat=json,location=46.86147689819336%2C-114.01863098144531
```

kali@kali: ~/Desktop/pcap3

```
┌──(kali㉿kali)-[~/Desktop/pcap3]
└─$ tshark -r "Capture 3.pcap" -Y "ip.src_host == "192.168.1.5" && http.request.method == "GET" && http.request.uri.query.parameter contains "location" && http.request.uri.path contains geocoding" -T fields -e http.request.uri.query.parameter > locations
```

Figure 13 - TShark command and Request URI Query Parameters

To parse this file properly and plot the points on a map, a python script was made. The python script and the relevant files can be found at **C2 – Capture 3/Program/**. After running the python script, an output was generated with the latitude and longitude coordinates cleaned up and a HTML file, which displayed a map with the markers in place.

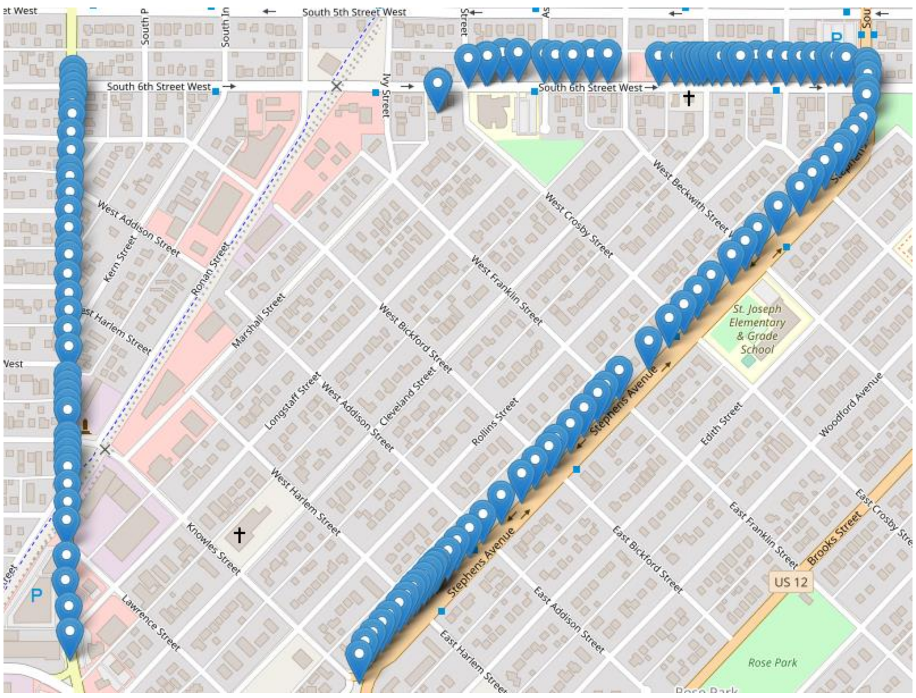


Figure 14 - Lat/Lng Coords on Map

It is clear that these markers are in the shape of the number 17, suggesting the date Ann and Kim Ill-song are meeting is the 17th of September at 5pm. Additionally, the top most point in the number 7 is slightly out of place and lands on a café. It is possible that this is the meeting location, however, this is just an assumption.

Critical Evaluation

The hardest part of this capture was finding the location data, as the investigator was looking for a date rather than a location. It was eventually found when the investigator tried searching key-words in Wiresharks 'Find Packet' function. In addition, it was initially confusing to retrieve the messages in the conversation between Ann and Castling. This was due to the capture picking up some of the messages multiple times but using the 'time' section in the JSON object made it easier to understand what order the messages were in.

Conclusions

Overall Critical Evaluation

Whilst exporting the data found in all three of the capture files, hashes of the files were taken to ensure that they remained unmodified during the analysis stage. Where possible, files were also stored in read-only mode to try to prevent any accidental modifications.

One of the mistakes initially made was where the recovered files were stored. As the investigator did not initially store them all in the same place, this made it difficult to analyse the files as they had to be found again to analyse them. This was fixed by creating a file for each capture and breaking them down into the headers that can be found in the appendixes. This made it much easier to analyse the files fully.

Reflection

There were several points in the investigation where the target had attempted to impede the investigation, some more succesfully than others.

The first capture did better in time-wasting than obfuscation, as opening all of the files is rather time consuming. However, it is fairly simple to identify Base64 encoding (this encoding often features padding with '==') and as all of the files used the same encoding, it was rather straight-forward to decode all of the files.

Fragmenting the image files found in the second capture by splitting the hex bytes and including the additional broken images, was an efficient way to throw off an investigator. If it wasn't for the tip off regarding the Edward Snowden quote, it is entirely possible that the investigator would not have been able to recover the image with the encoded message.

The final capture did not feature much obfuscation, however it would have been possible to hide these packets if the target had used the HTTPS protocol, rather than the HTTP protocol as this protocol is encrypted and requires verification.

References

Folium, 2013. *Folium 0.12.1 documentation.* [Online]
Available at: https://python-visualization.github.io/folium/index.html
[Accessed 1 December 2022].

Kumar, K., Sofat, S., Jain, S. & Aggarwal, N., 2012. Significance of Hash Value Generation in Digital Forensic: A Case Study. *International Journal of Engineering Research and Development,* 2(5), pp. 64-70.

Offensive Security, 2022. *Kali.* [Online]
Available at: https://www.kali.org/get-kali/#kali-installer-images
[Accessed 1 December 2022].

Wireshark, 2020. *SMB.* [Online]
Available at: https://wiki.wireshark.org/SMB
[Accessed 1 December 2022].

Wireshark, 2022. *tshark(1).* [Online]
Available at: https://www.wireshark.org/docs/man-pages/tshark.html
[Accessed 1 December 2022].

Appendix

Some files may appear to be missing, however this is because they did not fit into the word doc (.zips etc). They can be found here.

Appendix A – Capture 1 (Documents.zip)

A1 – Capture 1/Actual Documents/

GoT Spoilers.docx

Encoded

Sm9uIFNub3cgYnVybnMgZG93biBXaW50ZXJmZWx-
sIChhZ2FpbikgYW5kIHRoZSBXYWxsLg0KDQpIb2RvciBraWxscy-
BUaGVvbi4NCg0KRGFlbmVyeXMgZ2V0cyBlYXRl-
biBieSBhIGRyYWdv
bi4NCg0KU3Rhbm5p
cyBmYWxscyBpbiBsb3ZlIHdpdGggVHlyaW9uLiANCg0KDQo=

Decoded

Jon Snow burns down Winterfell (again) and the Wall.

Hodor kills Theon.

Daenerys gets eaten by a dragon.

Stannis falls in love with Tyrion.

NorthKorea.docx

Encoded

0JTQu9GPINC60L7Qs9C+INGN0YLQviDQvNC+0LbQtdGCINC60LDRgdCw0YLRjNGB
0Y86IA0KDQrQryDQsdGL0Lsg0YHQstC40LTQtdGC0LXQu9C10LwsINGH0YLQviDQ
mtC40Lwg0KfQtdC9INCj0L0g0Lgg0L/RgNCw0LLQuNGC0LXQu9GM0YHR-
gtCy0L4g0KHQtdCy
0LXRgNC90L7QuSDQmtC+0YDQtdC4INGA0LDQt9GA0LDQsdC+0YLQsNC70Lgg0L/
RgNC+0LPRgNCw0LzQvNG-
DLCDQutC+0YLQvtG
A0LDRjyDQv9C+0Lf
QstC+0LvRj9C10YIg0LjQvCDQv9GD0YLQtdGI0LXRgdGC0LLQvtCy0LDRgtG-
MINCy0L4g0LLRgNC10LzQtdC90LguINChINC40YHQv9C+0LvRjNC30L7QstCw0L3Q
uNC10Lwg0Y3RgtC+0Lkg0YLQtdGF0L3QvtC70L7Qs9C40LgsING-
PINGB0YfQuNGC0LDRjiwg0YfRgtC+INC+0L3QuCDQvdCw0LzQtdGA0LXQvdG-
LINC00LLQuNCz0LD
RgtGM0YHRjyDQstC
/0LXRgNC10LQg0Lgg0LjQt9C80LXQvdC40YLRjCDRgNC10LfRg9C70Yz-
RgtCw0YLRiyDQstC+0LnQvdG-
LINCyINCa0L7RgNC
10LUuIA0KDQrQn9C
+0LbQsNC70YPQudGB0YLQsCwg0J7QsdC4LdCS0LDQvSwg0YLRi-
yDQvNC+0Y8g0LXQtNC40L3RgdGC0LLQtdC90L3QsNGPINC90LDQtNC10LbQtNCw
Lg0KDQo=

Decoded

Для кого это может касаться:

Я был свидетелем, что Ким Чен Ун и правительство Северной Кореи разработали программу, которая позволяет им путешествовать во времени. С использованием этой технологии, я считаю, что они намерены двигаться вперед и изменить результаты войны в Корее.

Пожалуйста, Оби-Ван, ты моя единственная надежда.

Translated to English:

To whom it may concern:

I have witnessed that Kim Jong Un and the North Korean government have developed a program that allows them to travel through time. With the use of this technology, I believe they intend to move forward and change the outcome of the Korean War.

Please, Obi-Wan, you are my only hope.

PiD.docx

Encoded

RGVhciBFZCwNCg0KWWVhaCBJIHRvdGFsbHkgdG9vayBvdmVyIG-
ZvciBQYXVsIGFmdGVyIGh-
lIGRpZWQgaW4g4oCZNjYuIFlvdSBnb3QgbWUuIEFzIHlvdS-
BjYW4gc2VlLCB3ZSBkb27igJl0IGV2ZW4gbG9vayB0aGF0IG11Y2ggYWxpa2U6DQo=

Before(Paul) After(Me)

IAkgCQ0KQmVmb3JlKFBhdWwpIAkJCQkJQWZ0ZXIoTWUpDQoNCld-
lIGFyZW7igJl0IGV2ZW4gdGhlIHNhbWUgaGVpZ2h0ISBXaGF0IGNhbiBJI-
HNheSwgcGVvcGxlIGFyZSBzdHVwaWQuDQoNCg0KVGhhbmtzIG-
ZvciB0aGUgaW5xdWlyeSwNCg0KV2lsbGlhbSBDYW1wYmVsbA0KKFBhdWwgTWN
DYXJ0bmV5KQ0K

Decoded

Dear Ed,

Yeah I totally took over for Paul after he died in '66. You got me. As you can see, we don't even look that much alike:

Before(Paul) After(Me)

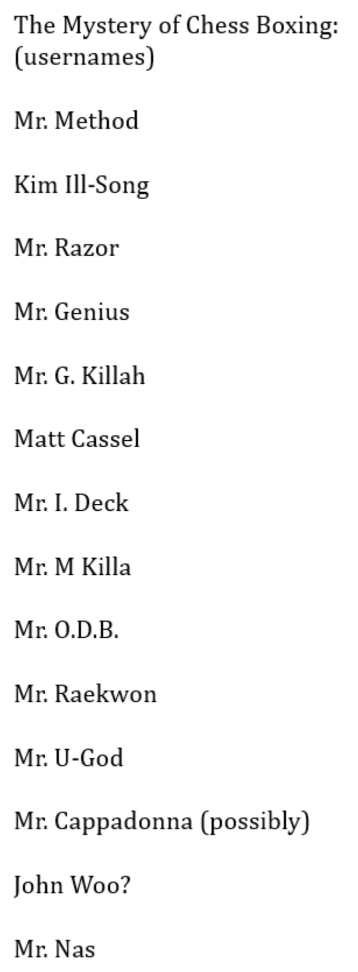
We aren't even the same height! What can I say, people are stupid.

Thanks for the inquiry,

William Campbell
(Paul McCartney)

A2 – Capture 1/Chess Boxing/
Chess Rules.docx

A3 – Capture1/Enter the WuTang/
track6.docx decoded



The Mystery of Chess Boxing:
(usernames)

Mr. Method

Kim Ill-Song

Mr. Razor

Mr. Genius

Mr. G. Killah

Matt Cassel

Mr. I. Deck

Mr. M Killa

Mr. O.D.B.

Mr. Raekwon

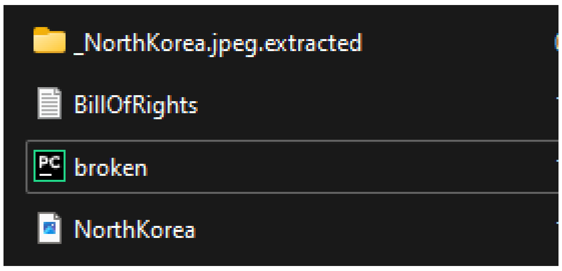
Mr. U-God

Mr. Cappadonna (possibly)

John Woo?

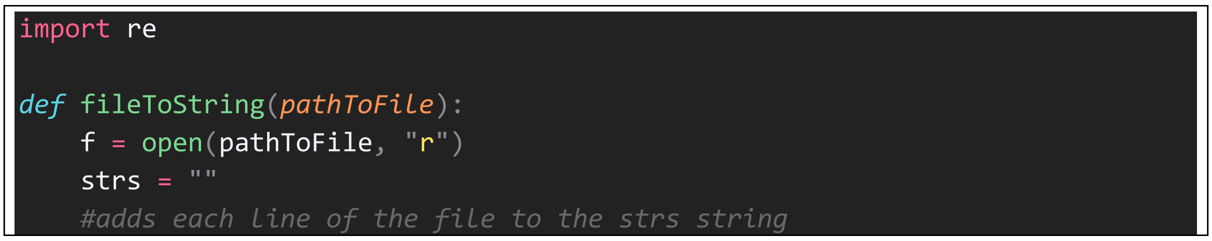
Mr. Nas

A4 – Capture1/More Documents/



_NorthKorea.jpeg.extracted
BillOfRights
broken
NorthKorea

A5 – Capture1/fixed.py

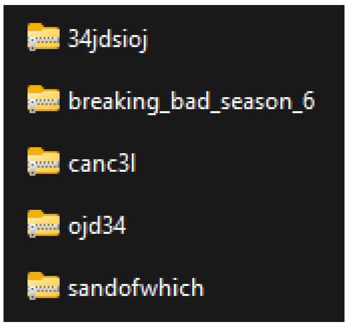


```
import re

def fileToString(pathToFile):
    f = open(pathToFile, "r")
    strs = ""
    #adds each line of the file to the strs string
    for line in f.readlines():
        strs+=line
    return strs
def ASCII(encoded):
    #number of ASCII characters
    NumOfASCII = len(encoded)
    #returns list of all ASCII characters
    return "".join([chr(i) for i in range(NumOfASCII)])
def sumName(name):
    sums=0
    #sums the indices in ASCII of all the characters in name
    for x in name:
        sums+=ord(x)
    return sums
def indexInFile(password):
    indices = []
    ASCIIArray = ASCII(encoded)
    #populates an array of indices to be used by the encoder
    for chrs in password:
        indices.append(ASCIIArray.index(chrs)+sumName(name)*2)
    return indices
def indexInASCII(name, encoded):
    indices = []
    ASCIIArray = ASCII(encoded)
    #split on all non-numeric characters
    #remove first index because it is blank
    indexList = re.split("[^\d]",encoded)[1:]
    #converts encoded characters to ASCII
    for index in indexList:
        indices.append(ASCIIArray[int(index) - (sumName(name)*2)])
    #returns decoded message
    return "".join(indices)
def encode(name):
    #returns a list of indices to be used for encoding
    indices = indexInFile(password,name)
    #convert file associated with name to a string
    bill = fileToString("./%s.txt"%name)
    encoded = ""
    #add letter in file plus index of the letter in the file to the encoded
string
    for index in indices:
        encoded+=bill[index]+str(index)

    return encoded


message="i2454 2497d2496n2502 2470 2500 2507o2436s2452
2500s2503n2502l2487e2456 2497 2500h2485l2487 2470b2490e2491a2501m2466
2483a2501a2501e2505 2497 2500a2486"
```

```
name="BillOfRights"

print(indexInASCII(name, message))
```

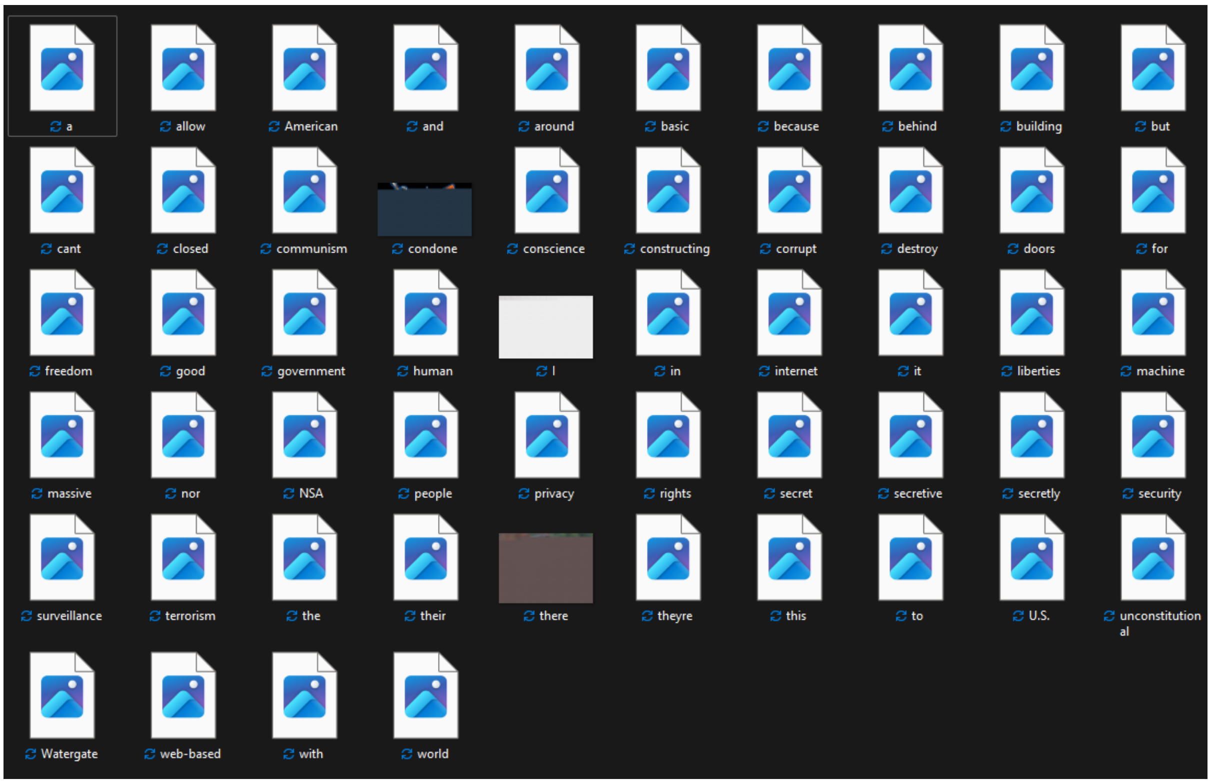
Appendix B – Capture 2

B1 – Capture 2/Zips/



34jdsioj
breaking_bad_season_6
canc3l
ojd34
sandofwhich

B2 – Capture 2/All photos/



a, allow, American, and, around, basic, because, behind, building, but,
cant, closed, communism, condone, conscience, constructing, corrupt, destroy, doors, for,
freedom, good, government, human, I, in, internet, it, liberties, machine,
massive, nor, NSA, people, privacy, rights, secret, secretive, secretly, security,
surveillance, terrorism, the, their, there, theyre, this, to, U.S., unconstitutional,
Watergate, web-based, with, world

B3 – Capture 2/final.jpg

Appendix C – Capture 3

C1 – Capture 3/Messages/

C2 – Capture 3/Program/

```python
import urllib.parse
import re
import folium

# decode URL-encoded string
def decode_url_encoding(encoded_string):
    return urllib.parse.unquote(encoded_string)

# split string after "location=" and return the second part
def split_location(string):
    return re.split(r"location=", string)[1]

# write string to a text file
def write_to_file(locations, map):
    with open("output.txt", "w") as f:
        for location in locations:
            f.write(location)
            folium.Marker(location.split(',')).add_to(map)
    return map

# main function
def main():
    locations = []
    map = folium.Map(zoom_start=8)
    # read list of data from file
    with open("locations") as f:
        data = f.readlines()

    # process each item in the list
    for item in data:
        # decode URL-encoding
        decoded_string = decode_url_encoding(item)

        # split after "location="
        locations.append(split_location(decoded_string))

    # write to file
    write_to_file(locations, map)

    map.save("map.html")

# run main function
if __name__ == "__main__":
    main()
```