CMP416 Assessment 2 - A Case of Bribery - Christopher Di-Nozzi, 1800317

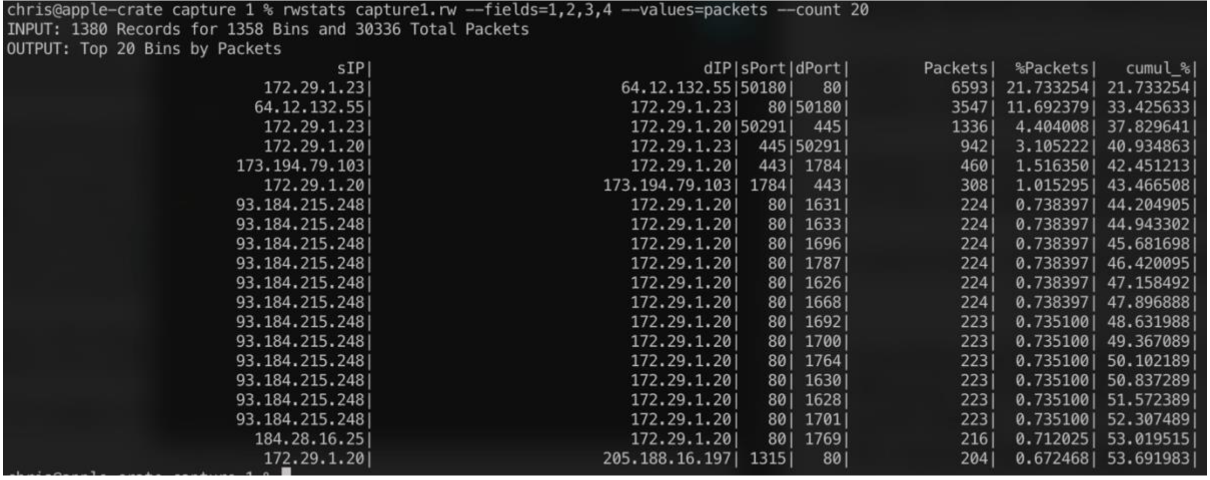
Table of Contents

Investigation of Capture1.pcap

This investigation began by conducting statistical flow analysis against the provided data. The file was first prepared for analysis following the steps outlined in Appendix A. Once prepared, the file was analysed using the 'rwstats' command, as part of the SiLK suite of network analyses tools, to examine the top 20 flows and display the source IP and port along with the destination IP and port. The following command was run to do this:

rwstats capture1.rw --fields=1,2,3,4 --values=packets --count 20

Fields 1,2,3, and 4 specified the inclusion of the source IP (sIP), destination IP (dIP), source port (sPort), and destination port (dPort). The value 'packets' sums all the packets across all records mapped to each bin. The output of this command can be seen in Figure 1



```
chris@apple-crate capture 1 % rwstats capture1.rw --fields=1,2,3,4 --values=packets --count 20
INPUT: 1380 Records for 1358 Bins and 30336 Total Packets
OUTPUT: Top 20 Bins by Packets
                       sIP|                             dIP|sPort|dPort|       Packets|  %Packets|    cumul_%|
               172.29.1.23|                    64.12.132.55|50180|   80|          6593| 21.733254| 21.733254|
              64.12.132.55|                     172.29.1.23|   80|50180|          3547| 11.692379| 33.425633|
               172.29.1.23|                     172.29.1.20|50291|  445|          1336|  4.404008| 37.829641|
               172.29.1.20|                     172.29.1.23|  445|50291|           942|  3.105222| 40.934863|
            173.194.79.103|                     172.29.1.20|  443| 1784|           460|  1.516350| 42.451213|
               172.29.1.20|                  173.194.79.103| 1784|  443|           308|  1.015295| 43.466508|
            93.184.215.248|                     172.29.1.20|   80| 1631|           224|  0.738397| 44.204905|
            93.184.215.248|                     172.29.1.20|   80| 1633|           224|  0.738397| 44.943302|
            93.184.215.248|                     172.29.1.20|   80| 1696|           224|  0.738397| 45.681698|
            93.184.215.248|                     172.29.1.20|   80| 1787|           224|  0.738397| 46.420095|
            93.184.215.248|                     172.29.1.20|   80| 1626|           224|  0.738397| 47.158492|
            93.184.215.248|                     172.29.1.20|   80| 1668|           224|  0.738397| 47.896888|
            93.184.215.248|                     172.29.1.20|   80| 1692|           223|  0.735100| 48.631988|
            93.184.215.248|                     172.29.1.20|   80| 1700|           223|  0.735100| 49.367089|
            93.184.215.248|                     172.29.1.20|   80| 1764|           223|  0.735100| 50.102189|
            93.184.215.248|                     172.29.1.20|   80| 1630|           223|  0.735100| 50.837289|
            93.184.215.248|                     172.29.1.20|   80| 1628|           223|  0.735100| 51.572389|
            93.184.215.248|                     172.29.1.20|   80| 1701|           223|  0.735100| 52.307489|
             184.28.16.25|                     172.29.1.20|   80| 1769|           216|  0.712025| 53.019515|
               172.29.1.20|                  205.188.16.197| 1315|   80|           204|  0.672468| 53.691983|
```

Figure 1: rwstats output of capture1.rw displaying the source port and IP along side the destination IP and port of the top 20 flows.

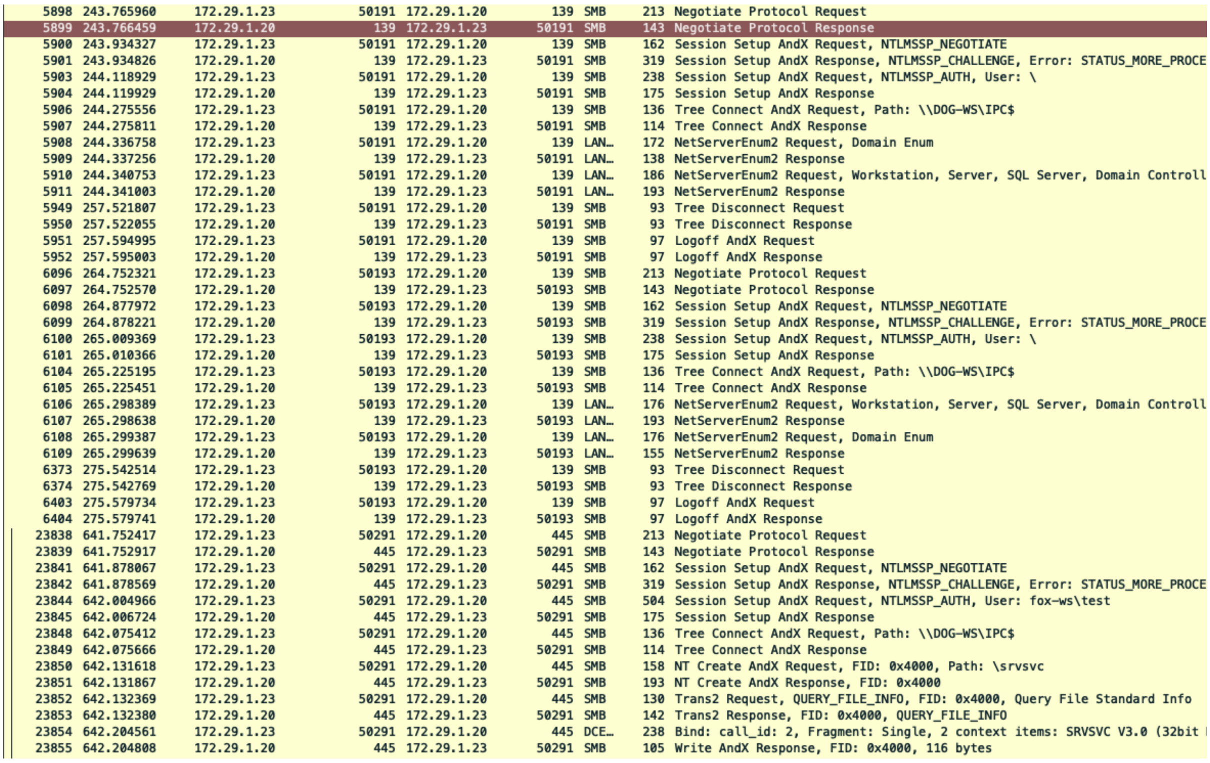
All the standard protocols seen (80/HTTP,445/SMB,443/HTTPS) could have been used for file transfers. HTTP and HTTPS could be used to download files from a web server, while SMB could have been used to transfer files from a network share. The first two flows (23->55 and 55->23) were examined but no traces of file transfers were found. Traffic related to AOL emails were found, and a zip file was uncovered which contained another .pcap file, but this was decided to be unrelated to the investigation at hand since it did not help to fulfil the given brief.

Analysing the SMB traffic provided significantly more interesting results. SMB is a remote file access protocol commonly used on Windows, (Wireshark, 2020). Therefore, it could be used to download files from one system to another.

The following query was used in Wireshark to filter the traffic between 172.29.1.23 and .20:

((ip.src==172.29.1.23 && ip.dst==172.29.1.20)||(ip.dst==172.29.1.23 && ip.src==172.29.1.20)) && smb

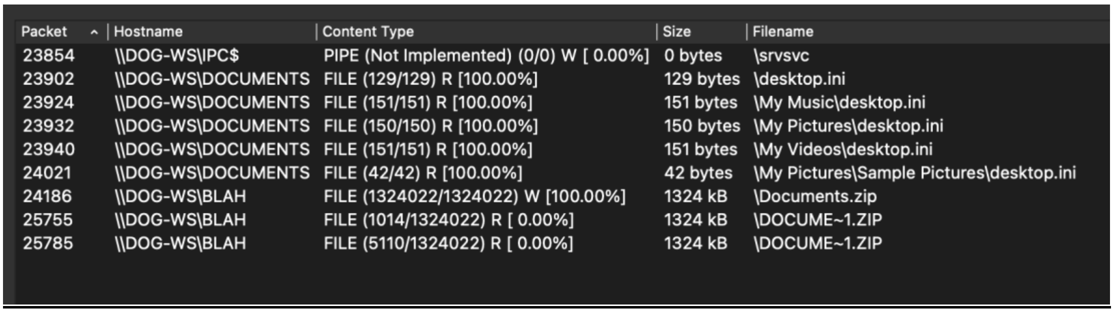
This query filtered all traffic that was from .23 towards .20, or vice versa. The '.src' and '.dst' parts were used to provide clarity to the reader. The "&& smb" keyword at the end caused only the SMB traffic to be displayed, rather than all the other associated TCP packets. This reduced the amount of noise displayed to the analyst and made finding key information easier. Analysing these packets showed there had been file transfers between the two network devices. A snippet of this traffic can be seen in Figure 2.



```
 5898 243.765960   172.29.1.23    50191 172.29.1.20      139 SMB    213 Negotiate Protocol Request
 5899 243.766459   172.29.1.20      139 172.29.1.23    50191 SMB    143 Negotiate Protocol Response
 5900 243.934327   172.29.1.23    50191 172.29.1.20      139 SMB    162 Session Setup AndX Request, NTLMSSP_NEGOTIATE
 5901 243.934826   172.29.1.20      139 172.29.1.23    50191 SMB    319 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCE
 5903 244.118929   172.29.1.23    50191 172.29.1.20      139 SMB    238 Session Setup AndX Request, NTLMSSP_AUTH, User: \
 5904 244.119929   172.29.1.20      139 172.29.1.23    50191 SMB    175 Session Setup AndX Response
 5906 244.275556   172.29.1.23    50191 172.29.1.20      139 SMB    136 Tree Connect AndX Request, Path: \\DOG-WS\IPC$
 5907 244.275811   172.29.1.20      139 172.29.1.23    50191 SMB    114 Tree Connect AndX Response
 5908 244.336758   172.29.1.23    50191 172.29.1.20      139 LAN…   172 NetServerEnum2 Request, Domain Enum
 5909 244.337256   172.29.1.20      139 172.29.1.23    50191 LAN…   138 NetServerEnum2 Response
 5910 244.340753   172.29.1.23    50191 172.29.1.20      139 LAN…   186 NetServerEnum2 Request, Workstation, Server, SQL Server, Domain Controll
 5911 244.341003   172.29.1.20      139 172.29.1.23    50191 LAN…   193 NetServerEnum2 Response
 5949 257.521807   172.29.1.23    50191 172.29.1.20      139 SMB     93 Tree Disconnect Request
 5950 257.522055   172.29.1.20      139 172.29.1.23    50191 SMB     93 Tree Disconnect Response
 5951 257.594995   172.29.1.23    50191 172.29.1.20      139 SMB     97 Logoff AndX Request
 5952 257.595003   172.29.1.20      139 172.29.1.23    50191 SMB     97 Logoff AndX Response
 6096 264.752321   172.29.1.23    50193 172.29.1.20      139 SMB    213 Negotiate Protocol Request
 6097 264.752570   172.29.1.20      139 172.29.1.23    50193 SMB    143 Negotiate Protocol Response
 6098 264.877972   172.29.1.23    50193 172.29.1.20      139 SMB    162 Session Setup AndX Request, NTLMSSP_NEGOTIATE
 6099 264.878221   172.29.1.20      139 172.29.1.23    50193 SMB    319 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCE
 6100 265.009369   172.29.1.23    50193 172.29.1.20      139 SMB    238 Session Setup AndX Request, NTLMSSP_AUTH, User: \
 6101 265.010366   172.29.1.20      139 172.29.1.23    50193 SMB    175 Session Setup AndX Response
 6104 265.225195   172.29.1.23    50193 172.29.1.20      139 SMB    136 Tree Connect AndX Request, Path: \\DOG-WS\IPC$
 6105 265.225451   172.29.1.20      139 172.29.1.23    50193 SMB    114 Tree Connect AndX Response
 6106 265.298389   172.29.1.23    50193 172.29.1.20      139 LAN…   176 NetServerEnum2 Request, Workstation, Server, SQL Server, Domain Controll
 6107 265.298638   172.29.1.20      139 172.29.1.23    50193 LAN…   193 NetServerEnum2 Response
 6108 265.299387   172.29.1.23    50193 172.29.1.20      139 LAN…   176 NetServerEnum2 Request, Domain Enum
 6109 265.299639   172.29.1.20      139 172.29.1.23    50193 LAN…   155 NetServerEnum2 Response
 6373 275.542514   172.29.1.23    50193 172.29.1.20      139 SMB     93 Tree Disconnect Request
 6374 275.542769   172.29.1.20      139 172.29.1.23    50193 SMB     93 Tree Disconnect Response
 6403 275.579734   172.29.1.23    50193 172.29.1.20      139 SMB     97 Logoff AndX Request
 6404 275.579741   172.29.1.20      139 172.29.1.23    50193 SMB     97 Logoff AndX Response
23838 641.752417   172.29.1.23    50291 172.29.1.20      445 SMB    213 Negotiate Protocol Request
23839 641.752917   172.29.1.20      445 172.29.1.23    50291 SMB    143 Negotiate Protocol Response
23841 641.878067   172.29.1.23    50291 172.29.1.20      445 SMB    162 Session Setup AndX Request, NTLMSSP_NEGOTIATE
23842 641.878569   172.29.1.20      445 172.29.1.23    50291 SMB    319 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCE
23844 642.004966   172.29.1.23    50291 172.29.1.20      445 SMB    504 Session Setup AndX Request, NTLMSSP_AUTH, User: fox-ws\test
23845 642.006724   172.29.1.20      445 172.29.1.23    50291 SMB    175 Session Setup AndX Response
23848 642.075412   172.29.1.23    50291 172.29.1.20      445 SMB    136 Tree Connect AndX Request, Path: \\DOG-WS\IPC$
23849 642.075666   172.29.1.20      445 172.29.1.23    50291 SMB    114 Tree Connect AndX Response
23850 642.131618   172.29.1.23    50291 172.29.1.20      445 SMB    158 NT Create AndX Request, FID: 0x4000, Path: \srvsvc
23851 642.131867   172.29.1.20      445 172.29.1.23    50291 SMB    193 NT Create AndX Response, FID: 0x4000
23852 642.132369   172.29.1.23    50291 172.29.1.20      445 SMB    130 Trans2 Request, QUERY_FILE_INFO, FID: 0x4000, Query File Standard Info
23853 642.132380   172.29.1.20      445 172.29.1.23    50291 SMB    142 Trans2 Response, FID: 0x4000, QUERY_FILE_INFO
23854 642.204561   172.29.1.23    50291 172.29.1.20      445 DCE…   238 Bind: call_id: 2, Fragment: Single, 2 context items: SRVSVC V3.0 (32bit
23855 642.204808   172.29.1.20      445 172.29.1.23    50291 SMB    105 Write AndX Response, FID: 0x4000, 116 bytes
```

Figure 2: A section of the SMB traffic analysed.

In Wireshark, by selecting File -> Export Objects -> SMB, a list of files transferred over SMB could be seen. In this list of 9 files seen in Figure 3. Of these files, only 1 was successfully downloaded, packet 24186, writing the file "Documents.zip" to the user's device.



Packet ^	Hostname	Content Type	Size	Filename
23854	\\DOG-WS\IPC$	PIPE (Not Implemented) (0/0) W [0.00%]	0 bytes	\srvsvc
23902	\\DOG-WS\DOCUMENTS	FILE (129/129) R [100.00%]	129 bytes	\desktop.ini
23924	\\DOG-WS\DOCUMENTS	FILE (151/151) R [100.00%]	151 bytes	\My Music\desktop.ini
23932	\\DOG-WS\DOCUMENTS	FILE (150/150) R [100.00%]	150 bytes	\My Pictures\desktop.ini
23940	\\DOG-WS\DOCUMENTS	FILE (151/151) R [100.00%]	151 bytes	\My Videos\desktop.ini
24021	\\DOG-WS\DOCUMENTS	FILE (42/42) R [100.00%]	42 bytes	\My Pictures\Sample Pictures\desktop.ini
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 3: Files transferred via SMB in Capture 1.

This transaction was confirmed to be between .23 and .20 by locating it within the above-mentioned Wireshark filter. The file could be extracted from the capture by selecting it from the form above and clicking save. Once saved, the zip could be extracted, and several files were revealed. The zip contained the following directories (d), sub-directories (sd) and files (f):

Documents (d)

- Actual Documents (sd)
  - GoT Spoilers.docx (f)
  - NotherKorea.docx (f)
  - PiD.docx (f)
- Chess Boxing (sd)
  - NK.jpg (f)
  - Rules 1..docx (f)
  - Rules 2.docx (f)

    - Rules 3.docx (f)
    - Rules 4.docx (f)
    - Rules 5.docx (f)
    - Rules 6.docx (f)
    - Rules 2.docx (f)
- Enter the WuTang(sd)
    - track6.docx (f)
    - track10.docx (f)
- More Documents(sd)
    - BillOfRights.txt (f)
    - NorthKorea.jpeg (f)

There was another zip file found within named "untitled.zip" which contained 5 other empty folders nested in each other. The final directory was named "SilentEye", the same name as a steganography tool. Using this hint, a python file was discovered within NorthKorean.jpeg but posed no use to the questions of this investigation.

All the above files were analysed. The contents of each .docx file was encoded using Base64. The file of most interest to this investigation was "track6.docx" found within "Enter the WuTang". After decoding this file, a list of usernames was discovered. This list could potentially be a list of actors involved in the bribery case. The full list can be found in Appendix B. A list of only the names has been provided below.

- 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

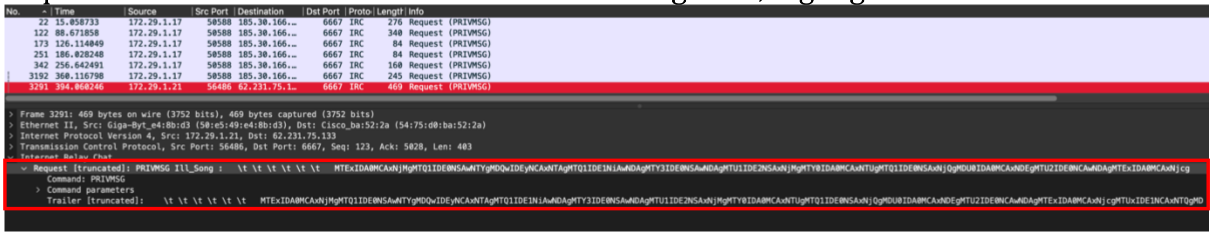
This analysis has shown the "Documents.zip" was transferred from 172.29.1.20 to 172.29.1.23. The recovery and decoding method have been outlined and the most likely contender for aliases of actors within the corruption case have been retrieved.

Investigation of Capture2.pcap

The brief for this capture mentioned IRC traffic, therefore, this was used as a starting point for analysis. IRC traffic could be filtered in Wireshark by simply filtering the keyword “IRC”. This could be filtered further to display only the requests that included private messages using the filter “irc.request.command==PRIVMSG”. There was IRC traffic from two different internal address, 172.29.1.17 and 172.29.1.21, however, packet number 22 from .17 contained the full conversation transaction and was therefore used for further analyses. Packet 22 could be right clicked and Follow -> TCP Protocol could be selected to view the entire flow. To automate the extraction of the messages, the tshark[1] tool was used with the following command line:

./tshark -r capture2.pcap -Y "(irc.request.command==PRIVMSG || irc.response.command==PRIVMSG) && (ip.addr ==172.29.1.17)" -T fields -e "irc.request" -e "irc.response”

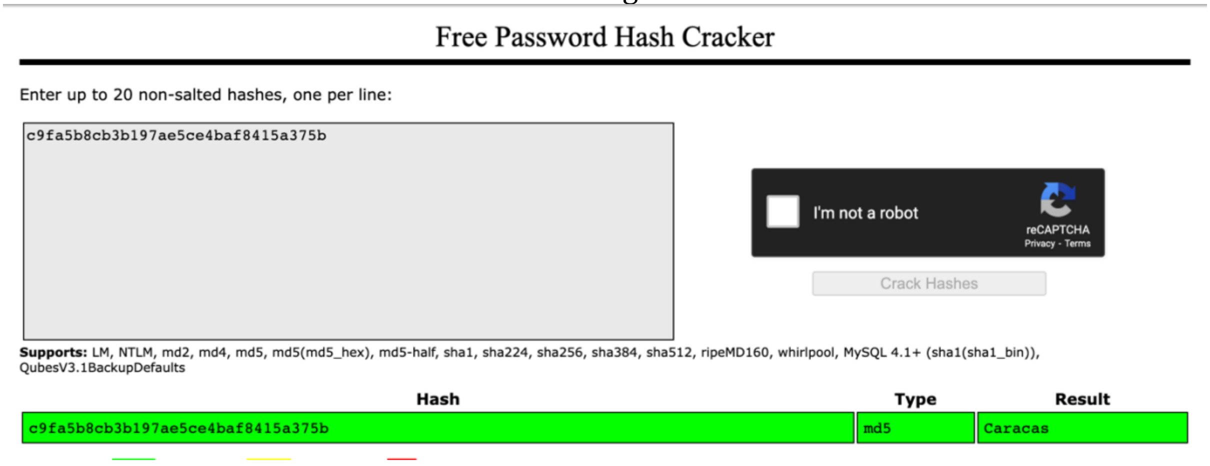
The above command invoked tshark against the capture2.pcap file provided for analysis. The capture was filtered from all IRC requests and responses that included the PRIVMSG command, and that were sent from or to the .17 IP address. These requests then had the request and response sections extracted from them. An example of a request section from Wireshark can be seen in Figure 4, highlighted in red.



```
No.   ^ Time        Source        Src Port Destination   Dst Port Proto Length Info
   22 15.058733    172.29.1.17     50588 185.30.166...     6667 IRC      276 Request (PRIVMSG)
  122 88.671858    172.29.1.17     50588 185.30.166...     6667 IRC      340 Request (PRIVMSG)
  173 126.114049   172.29.1.17     50588 185.30.166...     6667 IRC       84 Request (PRIVMSG)
  251 186.028248   172.29.1.17     50588 185.30.166...     6667 IRC       84 Request (PRIVMSG)
  342 256.642491   172.29.1.17     50588 185.30.166...     6667 IRC      160 Request (PRIVMSG)
 3192 360.116798   172.29.1.17     50588 185.30.166...     6667 IRC      245 Request (PRIVMSG)
 3291 394.060246   172.29.1.21     56486 62.231.75.1...    6667 IRC      469 Request (PRIVMSG)

> Frame 3291: 469 bytes on wire (3752 bits), 469 bytes captured (3752 bits)
> Ethernet II, Src: Giga-Byt_e4:8b:d3 (50:e5:49:e4:8b:d3), Dst: Cisco_ba:52:2a (54:75:d0:ba:52:2a)
> Internet Protocol Version 4, Src: 172.29.1.21, Dst: 62.231.75.133
> Transmission Control Protocol, Src Port: 56486, Dst Port: 6667, Seq: 123, Ack: 5028, Len: 403
v Internet Relay Chat
  v Request [truncated]: PRIVMSG Ill_Song :   \t \t \t \t \t \t   MTExIDA0MCAxNjMgMTQ1IDE0NSAwNTYgMDQwIDEyNCAxNTAgMTQ1IDE1NiAwNDAgMTY3IDE0NSAwNDAgMTU1IDE2NSAxNjMgMTY0IDA0MCAxNTUgMTQ1IDE0NSAxNjQgMDU0IDA0MCAxNDEgMTU2IDE0NCAwNDAgMTExIDA0MCAxNjcg
      Command: PRIVMSG
    > Command parameters
      Trailer [truncated]:   \t \t \t \t \t \t   MTExIDA0MCAxNjMgMTQ1IDE0NSAwNTYgMDQwIDEyNCAxNTAgMTQ1IDE1NiAwNDAgMTY3IDE0NSAwNDAgMTU1IDE2NSAxNjMgMTY0IDA0MCAxNTUgMTQ1IDE0NSAxNjQgMDU0IDA0MCAxNDEgMTU2IDE0NCAwNDAgMTExIDA0MCAxNjcgMTUxIDE1NCAxNTQgMD
```

Figure 4: An example of the request portion of the IRC packet, containing an encoded PRIVMSG.

The contents of the above command line were outputted into a text file. Once extracted, the conversation appeared to be encoded. Messages from Ill Song were decoded first from Base64 and then Base32. Messages from Razor were first decoded from Base64 and then Hex. Messages from Genius were decoded using Base64 and then Octal. One message from Genius contained an MD5 hash that was cracked separately using crackstation[2]. This can be seen in the below figure.



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
c9fa5b8cb3b197ae5ce4baf8415a375b
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
c9fa5b8cb3b197ae5ce4baf8415a375b	md5	Caracas

Figure 5: Cracked hash found within the IRC conversation between Ill Song and Genius.

[1] https://www.Wireshark.org/docs/man-pages/tshark.html

[2] https://crackstation.net/

Messages from Method were decoded via Base64 and then Hex. The second message sent to Method was unusually decoded. After being decoded from Base64, the first half ("SSBhbSBqdXN0IGhvcGVmdWwuIEl0IHdvdWxkIG1lYW4gc28gbXVjaCB0byBoYXZlIHR ") of the messages could again be Base64 decoded into an incomplete message: "I am just hopeful. It would mean so much to have t". The second half of the original message could be Base32 decoded to show the whole message: "I am just hopeful. It would mean so much to have the Title here. Please consider it.". It's unclear whether this was a technical issue or a obfuscation tactic.

Messages from Killah were decoded using Base64 and then Octal decoding. Finally, messages from Raekwon were decoded using Base64 and then Hex. All decoding was done using CyberChef[3]. The table below lays out the various general techniques used by each actor to encode their own messages.

Table 1: The general technique used by each actor to encode their IRC messages.

Technique	Actor
Base32 -> Base64	Ill Song
Hex -> Base64	Razor, Raekwon, Method
Octal -> Base64	Genius, Killah

The fully decoded conversation can be found in **Appendix C**.

From the conversations, claims can be made about the location and innocence of all parties to some extent. This information has been condensed into a table for easy comprehension.

Table 2: The Country and Conviction of each actor involved in the IRC conversations.

Actor	Country	Status
Razor	Paris, France	Guilty
Genius	Likely Caracas, Venezuela	Likely Guilty
Method	Unknown	Innocent
Killah	Qatar	Innocent
Raekwon	Russia	Guilty

Razor was assumed to be in Paris due to the mention of "The City of Love" by Ill Song. Razor also accepted a bribe of $700,000 after a short period negotiation.

Genius is assumed to be in Caracas due to its mention as a meeting point in their conversation, however, it could be the case that this was simply a meeting point and not their own country. From the conversation analysed, it's unclear whether Genius is guilty as no bribe is explicitly accepted, however, due to their intention to "see the validity of this claim" it could be assumed there is intention to accept a bribe if presented.

Method's location could not be pinpointed using the conversation, but made it clear they had no intention of speaking to Ill Song and took no bribe.

Killah is most likely located in Qatar since Ill Song asks about the weather there, and Killah answers "Hot, as always.", implying they have been there a while. Killah is seemingly not guilty of bribery, stating to Ill Song "We do not take kindly to this pathetic notion of bribery."

Finally, Raekwon appears to be based in Russia due to the discussion of a payment being made in Rubles, the currency used in Russia. Raekwon demands a bribe of 20

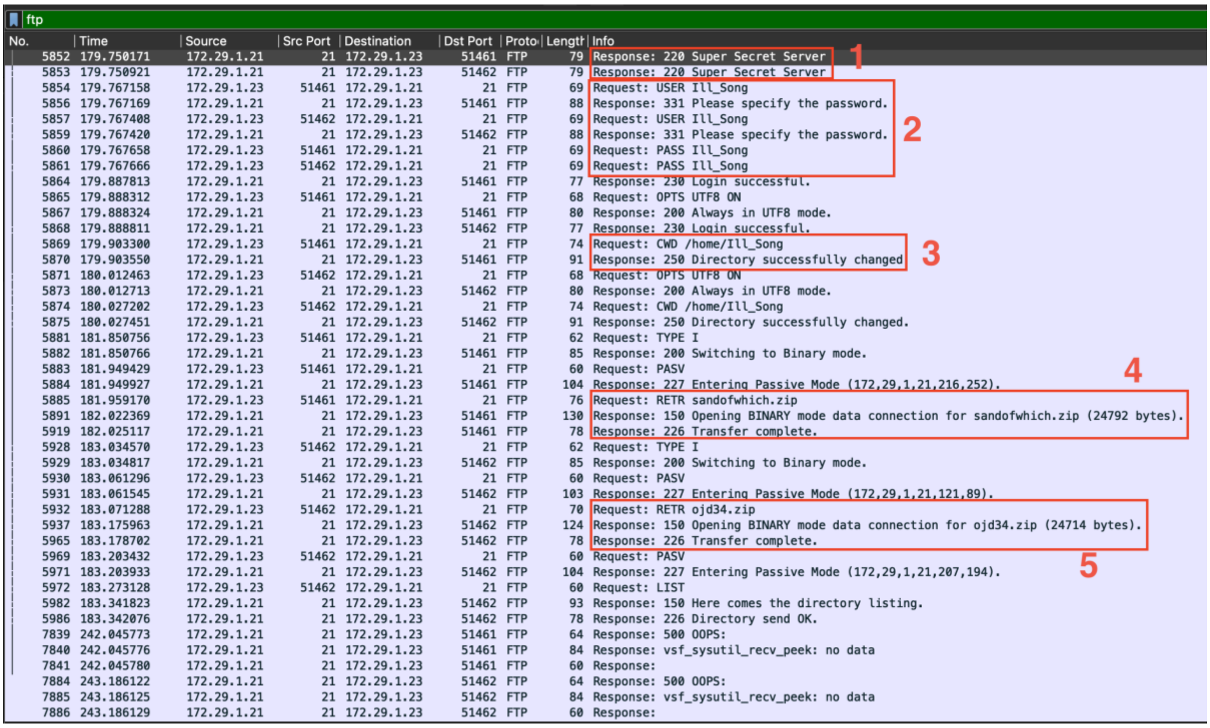
[3] https://gchq.github.io/CyberChef/

million Rubles and is told by Ill Song that it will be delivered. Thus, Raekwon is also guilty of bribery.

Investigation of Capture3.pcap

The brief for this capture highlighted FTP traffic, therefore, this is where the investigation began. FTP, or File Transfer Protocol, is used to transfer files, usually from a dedicated file storage server to a client but can also be used between two clients who host the server themselves.

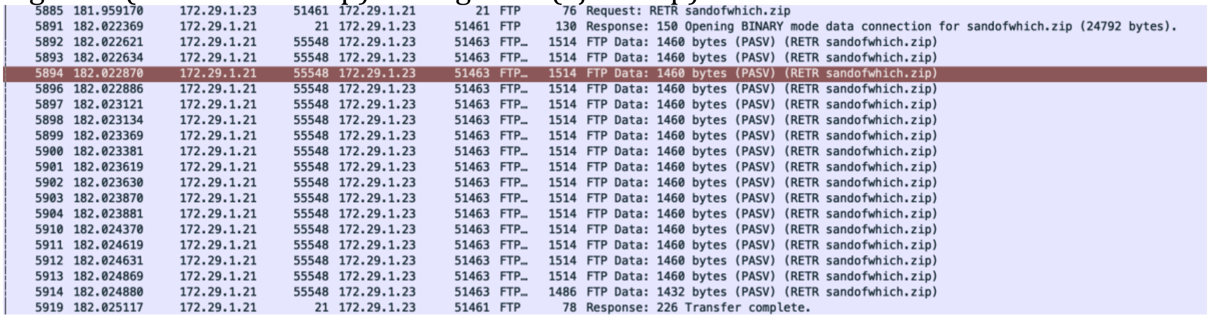
To filter for FTP in Wireshark, FTP can simply be entered in the filter bar. This displayed FTP traffic between 172.29.1.21 and 172.29.1.23. .23 appeared to be the server, nicknamed “Super Secret Server”, as can be seen in its initial response to the request from .21 (1). The user then logged in with the username and password “Ill_Song” (2) and proceeded to change into the “/home/Ill_Song” directory (3). Then the “sandofwhich.zip” (4) and “ojd34.zip” (5) files were transferred successfully from the server to the client. All the above cations can be seen in Figure 6.



```
ftp
No.   Time        Source       Src Port Destination  Dst Port Proto Length Info
 5852 179.750171  172.29.1.21        21 172.29.1.23    51461 FTP      79 Response: 220 Super Secret Server
 5853 179.750921  172.29.1.21        21 172.29.1.23    51462 FTP      79 Response: 220 Super Secret Server
 5854 179.767158  172.29.1.23     51461 172.29.1.21       21 FTP      69 Request: USER Ill_Song
 5856 179.767169  172.29.1.21        21 172.29.1.23    51461 FTP      88 Response: 331 Please specify the password.
 5857 179.767408  172.29.1.23     51462 172.29.1.21       21 FTP      69 Request: USER Ill_Song
 5859 179.767420  172.29.1.21        21 172.29.1.23    51462 FTP      88 Response: 331 Please specify the password.
 5860 179.767658  172.29.1.23     51461 172.29.1.21       21 FTP      69 Request: PASS Ill_Song
 5861 179.767666  172.29.1.23     51462 172.29.1.21       21 FTP      69 Request: PASS Ill_Song
 5864 179.887813  172.29.1.21        21 172.29.1.23    51461 FTP      77 Response: 230 Login successful.
 5865 179.888312  172.29.1.23     51461 172.29.1.21       21 FTP      68 Request: OPTS UTF8 ON
 5867 179.888324  172.29.1.21        21 172.29.1.23    51461 FTP      80 Response: 200 Always in UTF8 mode.
 5868 179.888811  172.29.1.21        21 172.29.1.23    51462 FTP      77 Response: 230 Login successful.
 5869 179.903300  172.29.1.23     51461 172.29.1.21       21 FTP      74 Request: CWD /home/Ill_Song
 5870 179.903550  172.29.1.21        21 172.29.1.23    51461 FTP      91 Response: 250 Directory successfully changed
 5871 180.012463  172.29.1.23     51462 172.29.1.21       21 FTP      68 Request: OPTS UTF8 ON
 5873 180.012713  172.29.1.21        21 172.29.1.23    51462 FTP      80 Response: 200 Always in UTF8 mode.
 5874 180.027202  172.29.1.23     51462 172.29.1.21       21 FTP      74 Request: CWD /home/Ill_Song
 5875 180.027451  172.29.1.21        21 172.29.1.23    51462 FTP      91 Response: 250 Directory successfully changed.
 5881 181.850756  172.29.1.23     51461 172.29.1.21       21 FTP      62 Request: TYPE I
 5882 181.850766  172.29.1.21        21 172.29.1.23    51461 FTP      85 Response: 200 Switching to Binary mode.
 5883 181.949429  172.29.1.23     51461 172.29.1.21       21 FTP      60 Request: PASV
 5884 181.949927  172.29.1.21        21 172.29.1.23    51461 FTP     104 Response: 227 Entering Passive Mode (172,29,1,21,216,252).
 5885 181.959170  172.29.1.23     51461 172.29.1.21       21 FTP      76 Request: RETR sandofwhich.zip
 5891 182.022369  172.29.1.21        21 172.29.1.23    51461 FTP     130 Response: 150 Opening BINARY mode data connection for sandofwhich.zip (24792 bytes).
 5919 182.025117  172.29.1.21        21 172.29.1.23    51461 FTP      78 Response: 226 Transfer complete.
 5928 183.034570  172.29.1.23     51462 172.29.1.21       21 FTP      62 Request: TYPE I
 5929 183.034817  172.29.1.21        21 172.29.1.23    51462 FTP      85 Response: 200 Switching to Binary mode.
 5930 183.061296  172.29.1.23     51462 172.29.1.21       21 FTP      60 Request: PASV
 5931 183.061545  172.29.1.21        21 172.29.1.23    51462 FTP     103 Response: 227 Entering Passive Mode (172,29,1,21,121,89).
 5932 183.071288  172.29.1.23     51462 172.29.1.21       21 FTP      70 Request: RETR ojd34.zip
 5937 183.175963  172.29.1.21        21 172.29.1.23    51462 FTP     124 Response: 150 Opening BINARY mode data connection for ojd34.zip (24714 bytes).
 5965 183.178702  172.29.1.21        21 172.29.1.23    51462 FTP      78 Response: 226 Transfer complete.
 5969 183.203432  172.29.1.23     51462 172.29.1.21       21 FTP      60 Request: PASV
 5971 183.203933  172.29.1.21        21 172.29.1.23    51462 FTP     104 Response: 227 Entering Passive Mode (172,29,1,21,207,194).
 5972 183.273128  172.29.1.23     51462 172.29.1.21       21 FTP      60 Request: LIST
 5982 183.341823  172.29.1.21        21 172.29.1.23    51462 FTP      93 Response: 150 Here comes the directory listing.
 5986 183.342076  172.29.1.21        21 172.29.1.23    51462 FTP      78 Response: 226 Directory send OK.
 7839 242.045773  172.29.1.21        21 172.29.1.23    51461 FTP      64 Response: 500 OOPS:
 7840 242.045776  172.29.1.21        21 172.29.1.23    51461 FTP      84 Response: vsf_sysutil_recv_peek: no data
 7841 242.045780  172.29.1.21        21 172.29.1.23    51461 FTP      60 Response:
 7884 243.186122  172.29.1.21        21 172.29.1.23    51462 FTP      64 Response: 500 OOPS:
 7885 243.186125  172.29.1.21        21 172.29.1.23    51462 FTP      84 Response: vsf_sysutil_recv_peek: no data
 7886 243.186129  172.29.1.21        21 172.29.1.23    51462 FTP      60 Response:
```

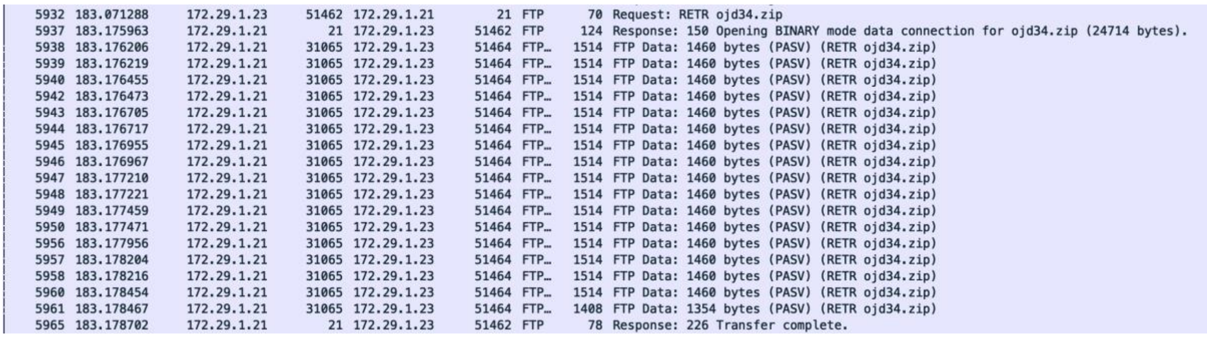
Figure 6: Annotated copy of the ftp traffic found in capture 3

To extract these two zip files, the analyst changed the filter from “ftp” to “ftp || ftp-data”, filtering to both ftp and ftp-data traffic. Ftp-data represents the data transported over port 20, rather than the port 21 most associated with FTP. Port 20 sends data, while port 21 handles the control. Once the filter was changed, the files could be extracted from their ftp-data streams. The two ftp-data streams containing the files can be seen in Figure 7 (sandofwhich.zip)and Figure 8 (ojd34.zip).



```
5885 181.959170  172.29.1.23  51461 172.29.1.21     21 FTP     76 Request: RETR sandofwhich.zip
5891 182.022369  172.29.1.21     21 172.29.1.23  51461 FTP    130 Response: 150 Opening BINARY mode data connection for sandofwhich.zip (24792 bytes).
5892 182.022621  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5893 182.022634  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5894 182.022870  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5896 182.022886  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5897 182.023121  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5898 182.023134  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5899 182.023369  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5900 182.023381  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5901 182.023619  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5902 182.023630  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5903 182.023870  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5904 182.023881  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5910 182.024370  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5911 182.024619  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5912 182.024631  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5913 182.024869  172.29.1.21  55548 172.29.1.23  51463 FTP… 1514 FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5914 182.024880  172.29.1.21  55548 172.29.1.23  51463 FTP… 1486 FTP Data: 1432 bytes (PASV) (RETR sandofwhich.zip)
5919 182.025117  172.29.1.21     21 172.29.1.23  51461 FTP     78 Response: 226 Transfer complete.
```

Figure 7: The ftp-data stream transferring sandofwhich.zip



```
  5932 183.071288      172.29.1.23     51462 172.29.1.21         21 FTP      70 Request: RETR ojd34.zip
  5937 183.175963      172.29.1.21        21 172.29.1.23      51462 FTP     124 Response: 150 Opening BINARY mode data connection for ojd34.zip (24714 bytes).
  5938 183.176206      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5939 183.176219      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5940 183.176455      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5942 183.176473      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5943 183.176705      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5944 183.176717      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5945 183.176955      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5946 183.176967      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5947 183.177210      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5948 183.177221      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5949 183.177459      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5950 183.177471      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5956 183.177956      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5957 183.178204      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5958 183.178216      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5960 183.178454      172.29.1.21     31065 172.29.1.23      51464 FTP…   1514 FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
  5961 183.178467      172.29.1.21     31065 172.29.1.23      51464 FTP…   1408 FTP Data: 1354 bytes (PASV) (RETR ojd34.zip)
  5965 183.178702      172.29.1.21        21 172.29.1.23      51462 FTP      78 Response: 226 Transfer complete.
```

Figure 8: The ftp-data stream transferring ojd34.zip

The files could then be extracted by right clicking on one of the data packets and selecting Follow -> TCP Stream. Then the “Show data as” format was changed to Raw and each file was saved under its respective name, “sandofwhich.zip” and “ojd34.zip”. Once extracted, both zips could be opened to reveal their contents. They each contained one folder full of .jpg file that could not be opened, except for “sandofwhich/I.jpg” which displayed nothing useful, as can be seen below.

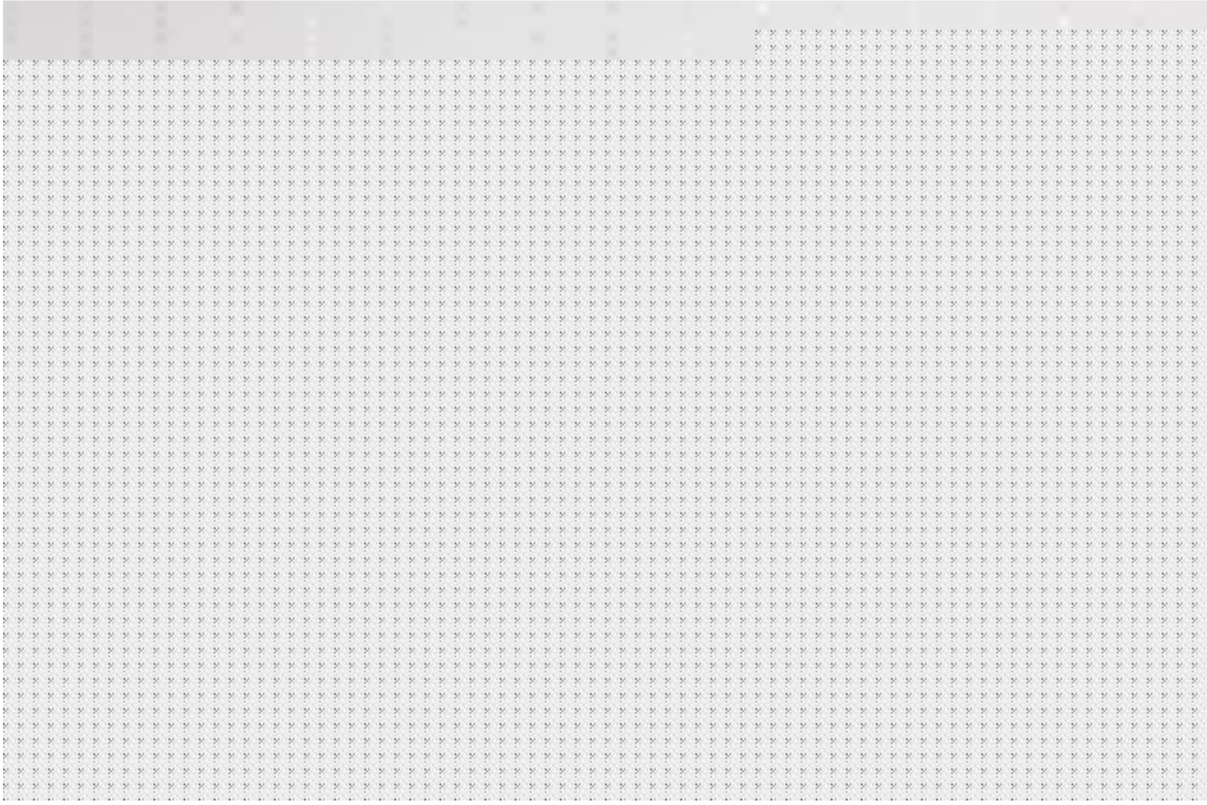
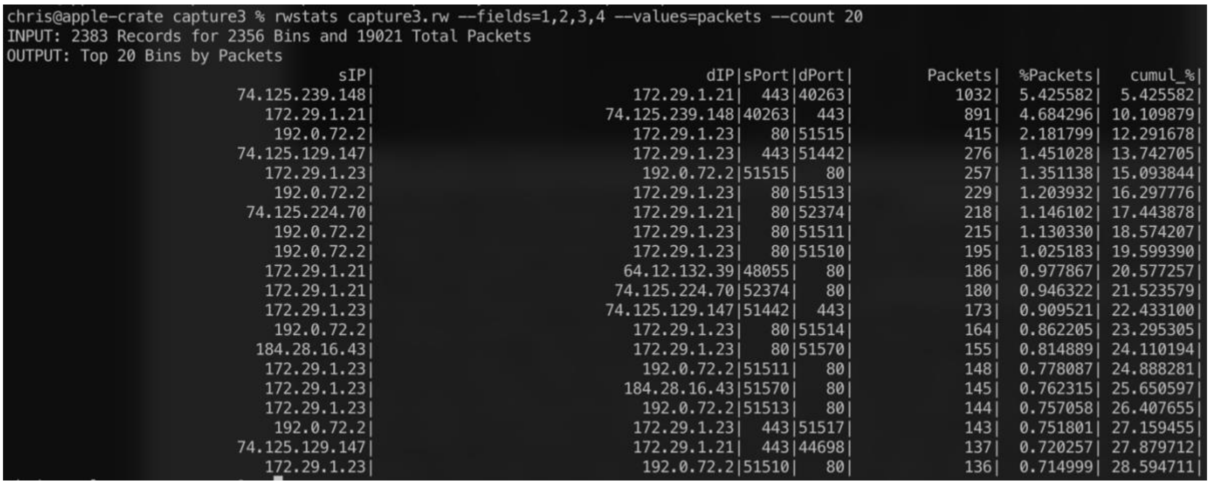


Figure 9: The contents of "sandofwhich/I.jpg"

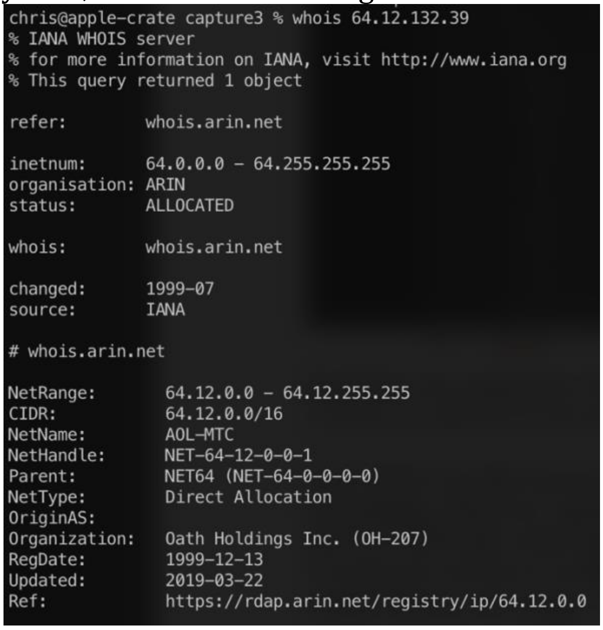
It appeared to be an incomplete image. All the files were named seemingly random words. At this stage, the hypothesis was that if a certain phrase was created with all the words, they could be combined into one image, however, there were not enough files between the two folders to create an Edward Snowden quote, as hinted too in the brief. Therefore, statistical flow analysis was employed to find more files.
Following the steps laid out in Appendix A, the file was prepared for analysis. The ‘rwstats’ command was then used to examine the top 20 flows in the traffic. The command and output can be seen in the figure below.



```
chris@apple-crate capture3 % rwstats capture3.rw --fields=1,2,3,4 --values=packets --count 20
INPUT: 2383 Records for 2356 Bins and 19021 Total Packets
OUTPUT: Top 20 Bins by Packets
                                    sIP|                                    dIP|sPort|dPort|             Packets| %Packets|   cumul_%|
                         74.125.239.148|                            172.29.1.21|  443|40263|                1032| 5.425582|  5.425582|
                            172.29.1.21|                         74.125.239.148|40263|  443|                 891| 4.684296| 10.109879|
                             192.0.72.2|                            172.29.1.23|   80|51515|                 415| 2.181799| 12.291678|
                         74.125.129.147|                            172.29.1.23|  443|51442|                 276| 1.451028| 13.742705|
                            172.29.1.23|                             192.0.72.2|51515|   80|                 257| 1.351138| 15.093844|
                             192.0.72.2|                            172.29.1.23|   80|51513|                 229| 1.203932| 16.297776|
                          74.125.224.70|                            172.29.1.21|   80|52374|                 218| 1.146102| 17.443878|
                             192.0.72.2|                            172.29.1.23|   80|51511|                 215| 1.130330| 18.574207|
                             192.0.72.2|                            172.29.1.23|   80|51510|                 195| 1.025183| 19.599390|
                            172.29.1.21|                           64.12.132.39|48055|   80|                 186| 0.977867| 20.577257|
                            172.29.1.21|                          74.125.224.70|52374|   80|                 180| 0.946322| 21.523579|
                            172.29.1.23|                         74.125.129.147|51442|  443|                 173| 0.909521| 22.433100|
                             192.0.72.2|                            172.29.1.23|   80|51514|                 164| 0.862205| 23.295305|
                           184.28.16.43|                            172.29.1.23|   80|51570|                 155| 0.814889| 24.110194|
                            172.29.1.23|                             192.0.72.2|51511|   80|                 148| 0.778087| 24.888281|
                            172.29.1.23|                           184.28.16.43|51570|   80|                 145| 0.762315| 25.650597|
                            172.29.1.23|                             192.0.72.2|51513|   80|                 144| 0.757058| 26.407655|
                             192.0.72.2|                            172.29.1.23|  443|51517|                 143| 0.751801| 27.159455|
                         74.125.129.147|                            172.29.1.21|  443|44698|                 137| 0.720257| 27.879712|
                            172.29.1.23|                             192.0.72.2|51510|   80|                 136| 0.714999| 28.594711|
```

Figure 10: rwstats command and output run against capture 3 as part of statistical flow analysis.

Each flow was analysed in Wireshark for any interesting traffic. The traffic between 172.29.1.21 and 64.12.132.39 proved most interesting. Performing a "whois" lookup against the 64.12.132.39 address revealed it was part of the ARIN CIDR block 64.12.0.0/16 owned by AOL, as can be seen in Figure 11.

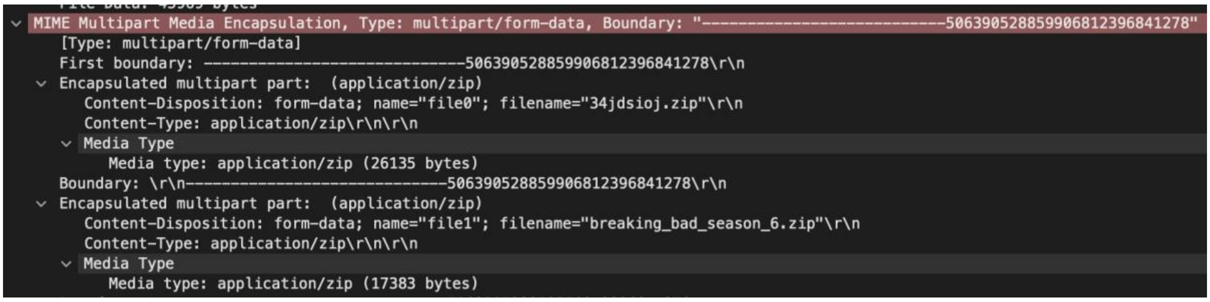


```
chris@apple-crate capture3 % whois 64.12.132.39
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

refer:        whois.arin.net

inetnum:      64.0.0.0 - 64.255.255.255
organisation: ARIN
status:       ALLOCATED

whois:        whois.arin.net

changed:      1999-07
source:       IANA

# whois.arin.net

NetRange:       64.12.0.0 - 64.12.255.255
CIDR:           64.12.0.0/16
NetName:        AOL-MTC
NetHandle:      NET-64-12-0-0-1
Parent:         NET64 (NET-64-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Oath Holdings Inc. (OH-207)
RegDate:        1999-12-13
Updated:        2019-03-22
Ref:            https://rdap.arin.net/registry/ip/64.12.0.0
```

Figure 11: Results of a whois lookup against the external IP 64.12.132.39

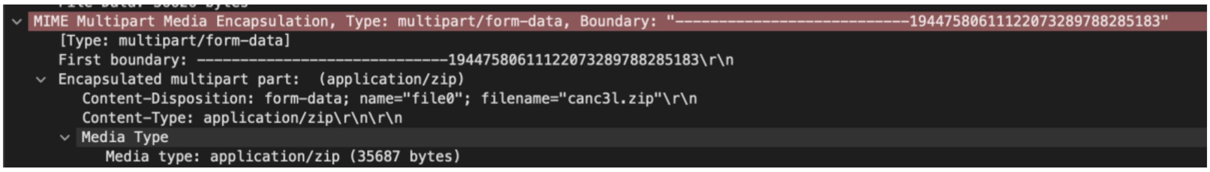
The traffic between the two devices appeared to be around AOLs email service. Two post requests were found that involved sending messages. These could be filtered in Wireshark using the following query:
((ip.dst==172.29.1.21 && ip.src==64.12.132.39) ||(ip.src==172.29.1.21 && ip.dst==64.12.132.39)) && http.request.method==POST && http.request.uri contains "SendMessage"

When sending an email via AOL, a "SendMessage" POST request is used. The above filter showed the traffic between the two Ips and then filtered again for POST requests and finally for POST requests containing "SendMessage". Both the packets found (2666 and 8190) contained zip files that could be extracted. These can be seen in Figure 12 and Figure 13.



```
v MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "---------------------------506390528859906812396841278"
    [Type: multipart/form-data]
    First boundary: -----------------------------506390528859906812396841278\r\n
  v Encapsulated multipart part:  (application/zip)
      Content-Disposition: form-data; name="file0"; filename="34jdsioj.zip"\r\n
      Content-Type: application/zip\r\n\r\n
    v Media Type
        Media type: application/zip (26135 bytes)
    Boundary: \r\n-----------------------------506390528859906812396841278\r\n
  v Encapsulated multipart part:  (application/zip)
      Content-Disposition: form-data; name="file1"; filename="breaking_bad_season_6.zip"\r\n
      Content-Type: application/zip\r\n\r\n
    v Media Type
        Media type: application/zip (17383 bytes)
```

Figure 12: The two zip files found in packet 2666



```
v MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "---------------------------194475806111220732897882851 83"
```

Figure 13: The one zip file found within 8190

All three zip files were extracted by right clicking the “Media type: application/zip” field and selecting “Export packet bytes”. Each file was saved as their respective name with the zip extension.

Once all five zip files had been extracted and unzipped, their contents first appeared meaningless. However, upon further examination, a select few files appeared to include the start of a JPEG file, noted by the hex characters “FF D8 FF”.

A script was written to combine all the files that made up an Edward Snowden quote and output them as one image. The quote that was the key to their assembly was “*I cant in good conscience allow the U.S. government to destroy privacy internet freedom and basic liberties for people around world with this massive surveillance machine theyre secretly building*”. This quote was found by searching the name of each valid JPEG file through Google until a quote that could be made using the words provided was found. Punctuation was removed from the quote along with the second instance of “the” between “for” and “people” since there was only one “the” file. The full script that was used to do this can be found in Appendix D.

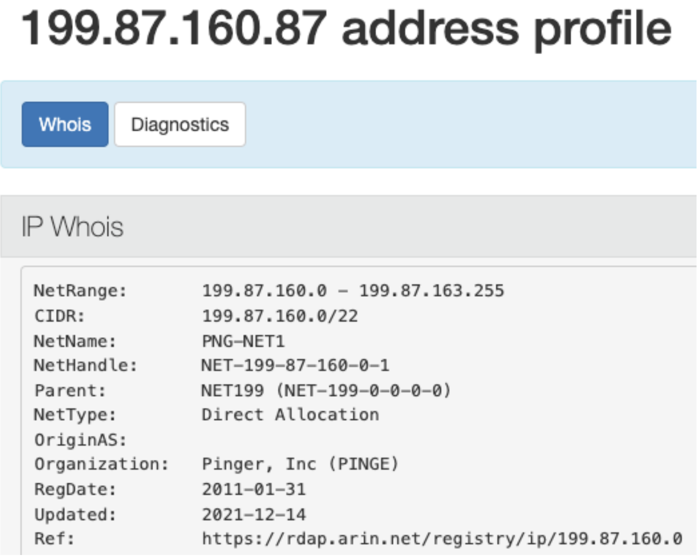
Once run, the script created an image of a futuristic chess board which can be seen below.

Figure 14: The image recovered from the combined 5 zip files found in capture 3

Anti-forensic techniques have clearly been used here, as this method of separating an image into coded file names amongst a vast number of red herrings it not a traditional way to transfer images. This behaviour should be treated as highly suspicious.

Investigation of Capture4.pcap

Due to the brief, IRC traffic was initially checked using Wireshark, but none was found. Statistical packet analysis was also employed but created no leads due to how thinly spread traffic appeared to be. HTTP traffic was searched manually via Wireshark and interesting traffic was found between 192.168.1.5 and 199.87.160.87. The latter IP belonged to "Pinger", a company that offers a messaging service, as can be seen in Figure 15.



199.87.160.87 address profile

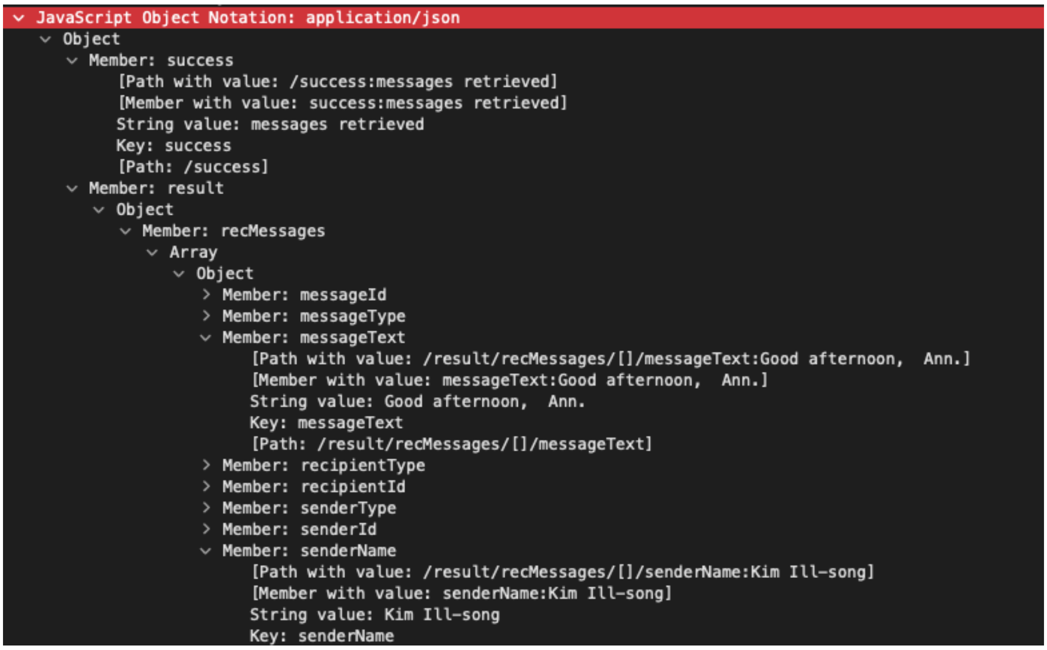
IP Whois

```
NetRange:       199.87.160.0 - 199.87.163.255
CIDR:           199.87.160.0/22
NetName:        PNG-NET1
NetHandle:      NET-199-87-160-0-1
Parent:         NET199 (NET-199-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Pinger, Inc (PINGE)
RegDate:        2011-01-31
Updated:        2021-12-14
Ref:            https://rdap.arin.net/registry/ip/199.87.160.0
```

Figure 15: A whois lookup against the server IP, revealing it as part of the block owned by Pinger.

The full interaction could be filtered using the following Wireshark query: *((ip.src==192.168.1.5 && ip.dst==199.87.160.87) || (ip.dst==192.168.1.5 && ip.src==199.87.160.87)) && http && http contains "messageText"*

This filter also only displayed the packets that had "messageText" in them. The data was sent using JSON files between the client and the server. These JSON files contained a "messageText" and "sendName" field which were critical in piecing together the conversation. An example JSON can be seen below.



```
⌄ JavaScript Object Notation: application/json
  ⌄ Object
    ⌄ Member: success
        [Path with value: /success:messages retrieved]
        [Member with value: success:messages retrieved]
        String value: messages retrieved
        Key: success
        [Path: /success]
    ⌄ Member: result
      ⌄ Object
        ⌄ Member: recMessages
          ⌄ Array
            ⌄ Object
              › Member: messageId
              › Member: messageType
              ⌄ Member: messageText
                  [Path with value: /result/recMessages/[]/messageText:Good afternoon,  Ann.]
                  [Member with value: messageText:Good afternoon,  Ann.]
                  String value: Good afternoon,  Ann.
                  Key: messageText
                  [Path: /result/recMessages/[]/messageText]
              › Member: recipientType
              › Member: recipientId
              › Member: senderType
              › Member: senderId
              ⌄ Member: senderName
                  [Path with value: /result/recMessages/[]/senderName:Kim Ill-song]
                  [Member with value: senderName:Kim Ill-song]
                  String value: Kim Ill-song
                  Key: senderName
```

Figure 16: An example JSON request sent from the server to the user containing message details.

Analysing the packets, the investigator found the conversation to be between two parties, Ann Decover and Kim Ill-Song. The conversation mentions a meeting in September at 5pm but does not specify the date. The entire conversation can be found in Appendix E.

To find the date, further HTTP traffic was analysed. Traffic was found to the domain "mob.mapquestapi.com". MapQuest is a mapping application, like Google Maps. Due to the high number of requests sent to the address and its unique nature, it was further investigated. All requests sent to the domain could be viewed using the following Wireshark filter:

http.host eq mob.mapquestapi.com

This traffic was then exported as text by selecting all the packets, clicking File -> Export Packet Dissections -> As plain text. This text file was then fed into a python script that extracted all the unique coordinates found inside the GET request and exported them into a CSV file. This script can be found in Appendix F. This CSV file was then fed into Google Maps to lay out the coordinates. Once laid out, the coordinates arranged into the number 17, presumably the date on which the meeting is to take place. This can be seen in the below figure.

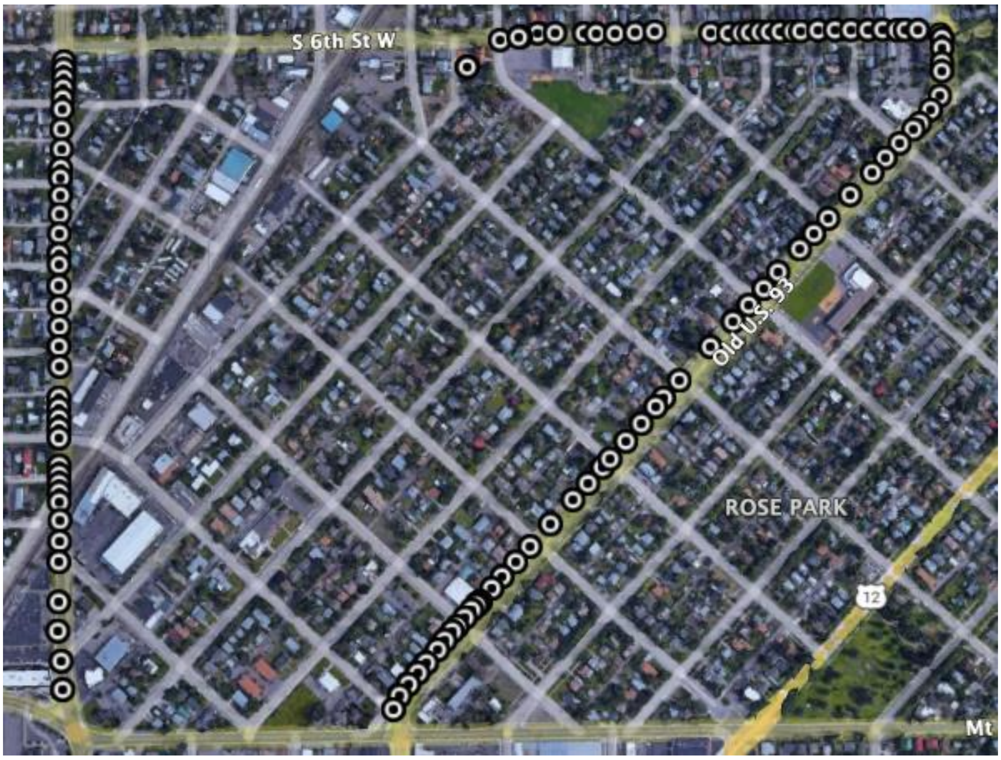


Figure 17: The coordinates found within the traffic laid out on Google Earth, spelling the number 17.

Therefore, it can be presumed that Ann Decover and Kim Ill-Song planned to meet on the 17th of September at 5pm.

Bibliography

Wireshark, 2020. *Server Message Block Protocol (SMB).* [Online]
Available at: https://wiki.wireshark.org/SMB
[Accessed 25 December 2021].

Appendix A

Statistical Flow Analysis File Preparation

To prepare a file for statistical flow analysis, the capture file was first converted into a "Yet Another Flowmeter" (YAF) file using the YAF tool. The following command was run against the capture PCAP file to do this:

```
yaf --in Capture.pcap --out capture.yaf
```

This converted the file from its original PCAP type to YAF. The file was then converted into the appropriate format to be analysed using the System for Internet-Level Knowledge (SiLK[4]). This was done by running the following command against the newly created YAF file:

```
rwipfix2silk capture.yaf --silk-output=capture.rw
```

This command generated a new .rw file that could then be used to conduct statistical flow analysis.

Appendix B
track6.docx

Encoded

VGhlIE15c3Rlcnkgb2YgQ2hlc3MgQm94aW5nOg0KKHVzZXJuYW1lcykNCg0KTXIuIE1ld
GhvZA0KDQpLaW0gSWxsLVNvbmcNCg0KTXIuIFJhem9yDQoNCk1yLiBHZW5pdXMNCg
0KTXIuIEcuIEtpbGxhaA0KDQpNYXR0IENhc3NlbA0KDQpNci4gSS4gRGVjaw0KDQpNci4
gTSBLaWxsYQ0KDQpNci4gTy5ELkIuDQoNCk1yLiBSYWVrd29uDQoNCk1yLiBVLUdvZA
0KDQpNci4gQ2FwcGFkb25uYSAocG9zc2libHkpDQoNCkpvaG4gV29vPw0KDQpNci4gT
mFzDQo=

Decoded

The Mystery of Chess Boxing:
(usernames)

Mr. Method

Kim Ill-Song

Mr. Razor

Mr. Genius

[4] https://tools.netsa.cert.org/silk/index.html

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

Appendix C
Decoded IRC Conversation

Ill Song: Mr. Razor, I am excited about the prospect of the Chess Boxing world title coming to Pyongyang.
Razor: Well the decision is not final yet.
Razor: I am a very busy man, but perhaps I could be persuaded to visit. See if Pyongyang is the right place for the World Title.
Ill Song : Perhaps not. How about I send you a gift? Something to get you out of the City of Love and take your own vacation somewhere.
Razor: Somewhere expensive, I hope.
Ill Song: 5
Razor: 9
Ill Song: 7
Razor: $700,000 it is. Where can I meet you?
Ill Song: I will be in touch with the address.

Ill Song: As we discussed earlier, I believe I might be able to help you with your search.

Genius: I see. Then we must meet, and I will see the validity of this claim.
Ill Song: I can be in Caracaswithin the week.
Genius: No. Not here. Can I not go to you?
Ill Song: I am afraid that would be unwise. I will send you a message with the date and location through a more secure form of communication.

Ill Song: Mr. Method, I am excited about the prospect of the Chess Boxing world title coming to Pyongyang.
Method: I am not sure who you are, but I have an idea. Either way, I am not interested.

Ill Song: I am just hopeful. It would mean so much to have the Title here. Please consider it.
Method: Do not speak to me again.

Ill Song: How is the weather in Qatar, Mr. Killah?
Killah: Hot, as always. Who is this?
Ill Song: I am a fan of Chess Boxing. I would love to see the Title held in Korea.
Killah: We will have to see how the bid turns out.
Ill Song: Is there anything that I could do to help make your decision easier?
Killah: No! The great nation of Qatar would never be swayed so easily.
Killah: Nor would I. We do not take kindly to this pathetic notion of bribery.

Ill Song: Mr. Raekwon, have you spoken with Mr. Razor?
Raekwon: I have, but I won.t be bought so easily.
Ill Song: Bought? Of course not. You are an official on the executive committee of the ICBA. I just want you to know that I am here to help make your decision as easy as possible.
Raekwon: I would need at least 20 million Rubles.
Ill Song: Consider it done. I will send you the information for the drop-off point soon.

Appendix D
Capture 3 Image Combination Script

```
quote = "I cant in good conscience allow the U.S. government to destroy
privacy internet freedom and basic liberties for people around world
with this massive surveillance machine theyre secretly building"

image = open("final.hex","wb")

for f in quote.split():
    f = f +".jpg"
    part = open(f,"rb")
    print("Handling: ",f)
    image.write(part.read())
```

Appendix E
Full Conversation

Kim Ill-song
Good afternoon, Ann.
Ann Dercover
who is this
Ann Dercover
where are you?
Kim Ill-song
Castling.
Ann Dercover
Do you know that there are people investigating Kim Ill-Song?
Kim Ill-song
I know I can't tell you that.

Kim Ill-song
Of course. However, they will never know it is me behind the bribes.
Ann Dercover
still we should be careful. Pay attention. I want to meet in September at 5PM.
Kim Ill-song
At our old meetup spot?
Ann Dercover
yes
Kim Ill-song
What day?
Ann Dercover
I told you to pay attention.

Appendix F

Python Script to extract coordinates

```
f = open("/Users/chris/Desktop/loc.txt", "r") #path to exported packets

lines = f.readlines()
coords=[]
for line in lines:
    if "location=" in line:
        str = line.split("location=",1)[1]
        #clean out junk
        str = str.replace("HTTP/1.1","")
        str = str.replace("%2C",",")
        str = str.replace("\\r\\n","")
        str = str.replace("\\n","")
        str = str.replace("]","")
        coords.append(str)

coords = list(set(coords)) #remove duplicates
print(coords)

#write to CSV
output = open("/Users/chris/Desktop/map.csv","w")
output.write("lat,lon\n")
for x in coords:
    output.write(x)
```