Network Forensics Investigation

Unit 2: Network Forensics

Ekku Jokinen

1703641

CMP416: Digital Forensics 2

BSc Ethical Hacking Year 3 (Accelerated)

2019/20

Note that Information contained in this document is for educational purposes.

.

Contents

.

1 INVESTIGATION OF CAPTURE 1.PCAP

1.1 ABSTRACT

A PCAP file was examined for any file transfers by exporting a ZIP file that was originally created containing files from an SMB network share. These documents included a text file, several Base64 encoded Microsoft Word documents and two image files and their contents included:

- TV show spoilers
- North Korean flag
- Rules to chess boxing
- Song lyrics
- U.S. Bill of Rights
- a list of usernames

One of the images had a hidden ZIP file inside it which contained a broken python script. One of the folders also hinted about a steganography tool used to hide messages in images but attempts to recover such messages from the images were unsuccessful.

1.2 PROCEDURE

The task was to find and identify any downloaded files from a PCAP, so the object export list for different types of traffic were analysed to find possible files of interest (Figure 1). There were several ZIP files transferred using a SMB network share but based on the file size and names they were most likely one file (Figure 2). The files were then exported and examined closer.

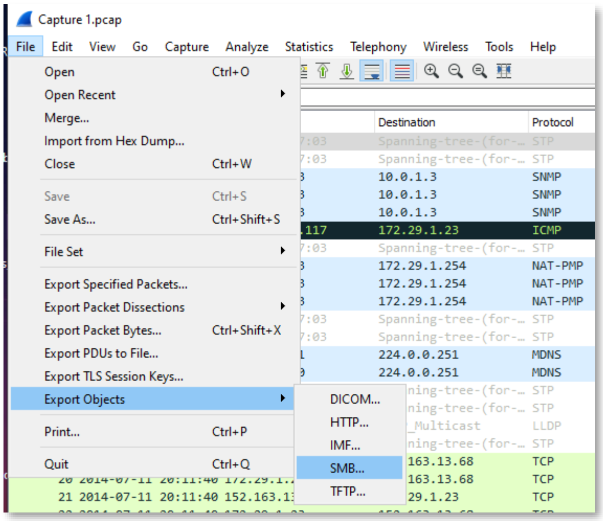
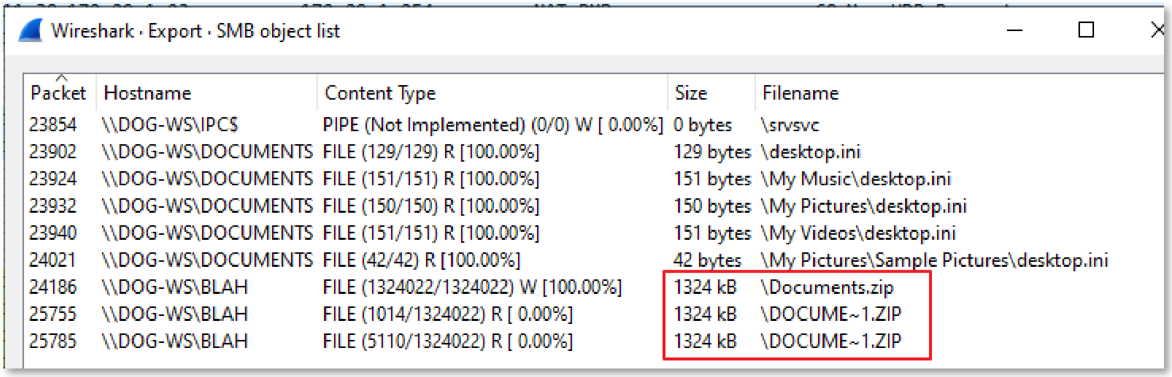


Figure 1

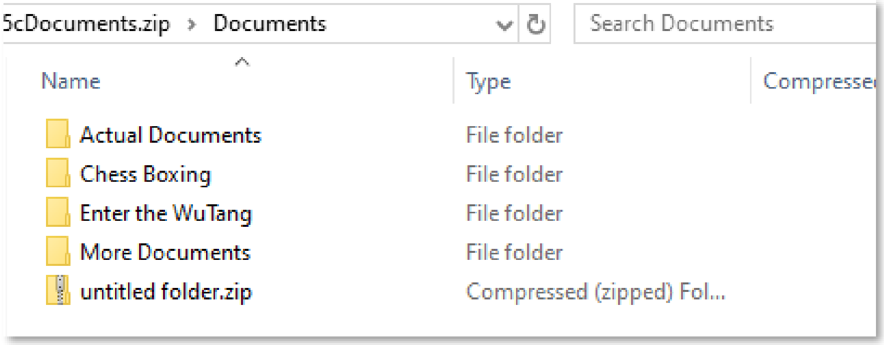


Wireshark · Export · SMB object list

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 2

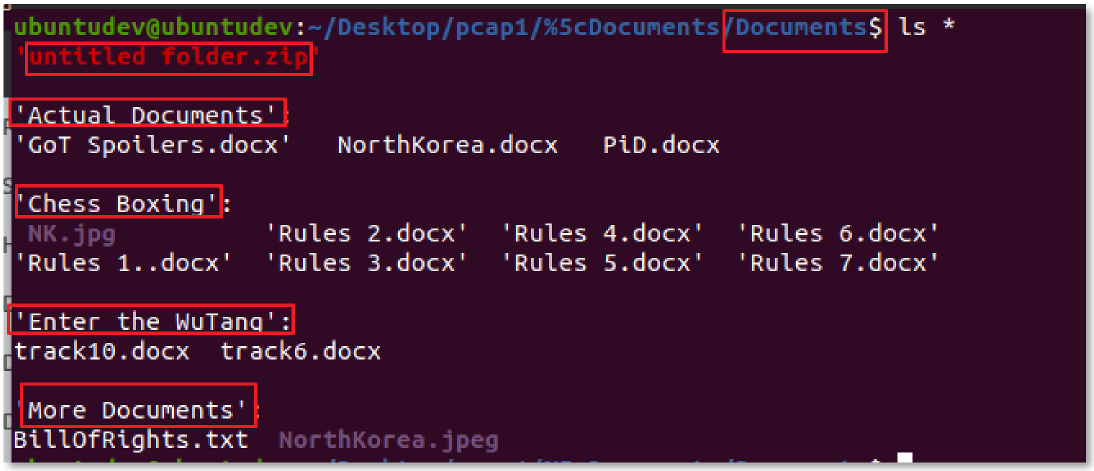
The ZIP file named Documents.zip had a folder called Documents inside which had four subfolders and one ZIP archive (Figure 3). The folders included several Microsoft Word documents, a text file and two jpeg images (Figure 4).



5cDocuments.zip › Documents

Name	Type	Compresse
Actual Documents	File folder	
Chess Boxing	File folder	
Enter the WuTang	File folder	
More Documents	File folder	
untitled folder.zip	Compressed (zipped) Fol...	

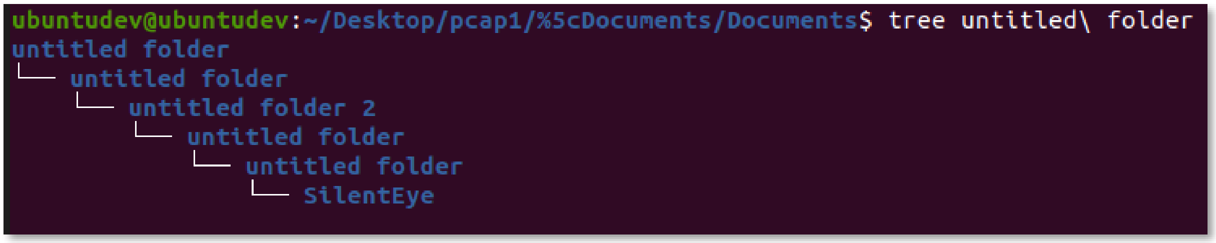
Figure 3



```
ubuntudev@ubuntudev:~/Desktop/pcap1/%5cDocuments/Documents$ ls *
'untitled folder.zip'

'Actual Documents':
'GoT Spoilers.docx'   NorthKorea.docx   PiD.docx

'Chess Boxing':
 NK.jpg            'Rules 2.docx'  'Rules 4.docx'  'Rules 6.docx'
'Rules 1..docx'   'Rules 3.docx'  'Rules 5.docx'  'Rules 7.docx'

'Enter the WuTang':
track10.docx  track6.docx

'More Documents':
BillOfRights.txt  NorthKorea.jpeg
```

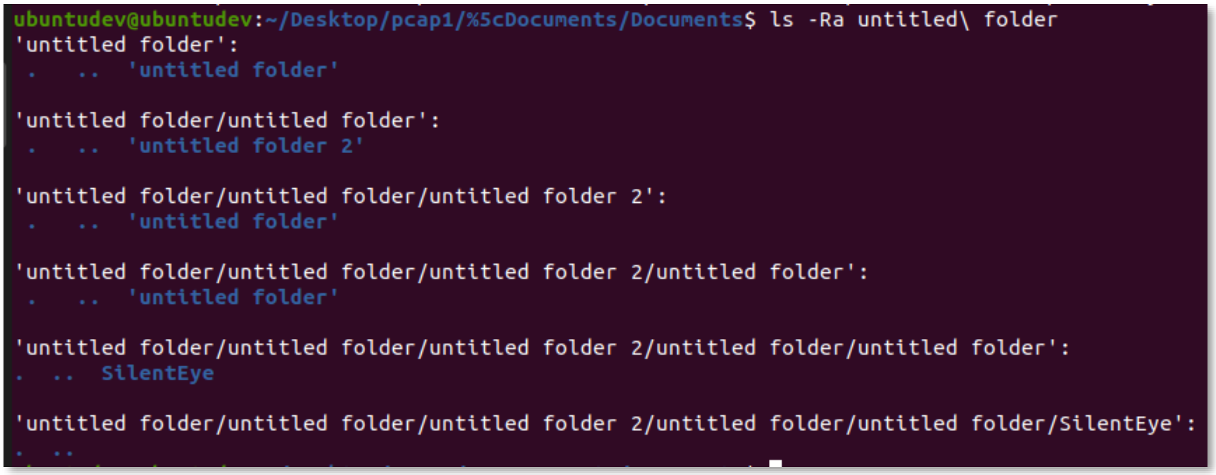
Figure 4

The ZIP archive (untitled folder.zip) had several subfolders inside one another, but all seemed empty (Figures 5 & 6). Only the final folder had a specific name, which made it seem significant and after an online search it was found that *SilentEye* was a steganography tool (Figure 7). However, there was nothing to indicate which file(s) it should be used with.



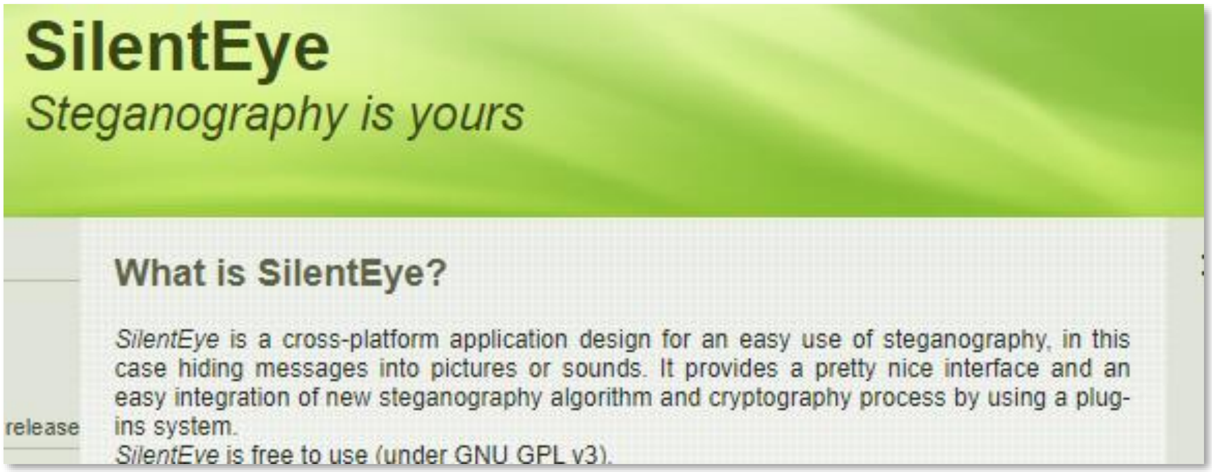
```
ubuntudev@ubuntudev:~/Desktop/pcap1/%5cDocuments/Documents$ tree untitled\ folder
untitled folder
└── untitled folder
    └── untitled folder 2
        └── untitled folder
            └── untitled folder
                └── SilentEye
```

Figure 5



```
ubuntudev@ubuntudev:~/Desktop/pcap1/%5cDocuments/Documents$ ls -Ra untitled\ folder
'untitled folder':
 .   ..  'untitled folder'

'untitled folder/untitled folder':
 .   ..  'untitled folder 2'

'untitled folder/untitled folder/untitled folder 2':
 .   ..  'untitled folder'

'untitled folder/untitled folder/untitled folder 2/untitled folder':
 .   ..  'untitled folder'

'untitled folder/untitled folder/untitled folder 2/untitled folder/untitled folder':
.  ..  SilentEye

'untitled folder/untitled folder/untitled folder 2/untitled folder/untitled folder/SilentEye':
.  ..
```

Figure 6



SilentEye

Steganography is yours

What is SilentEye?

SilentEye is a cross-platform application design for an easy use of steganography, in this case hiding messages into pictures or sounds. It provides a pretty nice interface and an easy integration of new steganography algorithm and cryptography process by using a plug-ins system.

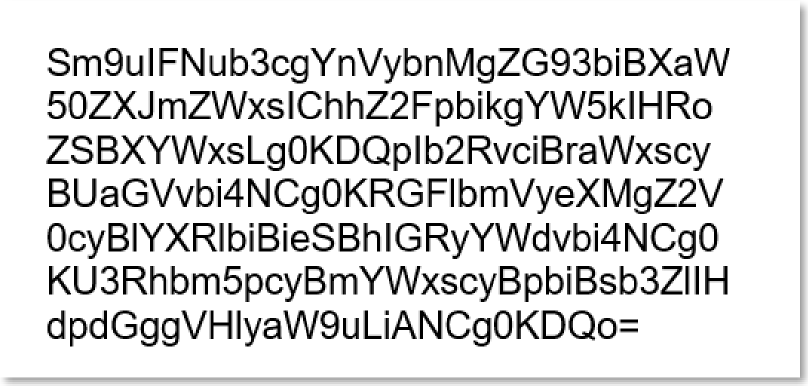
SilentEye is free to use (under GNU GPL v3).

Figure 7

Actual Documents

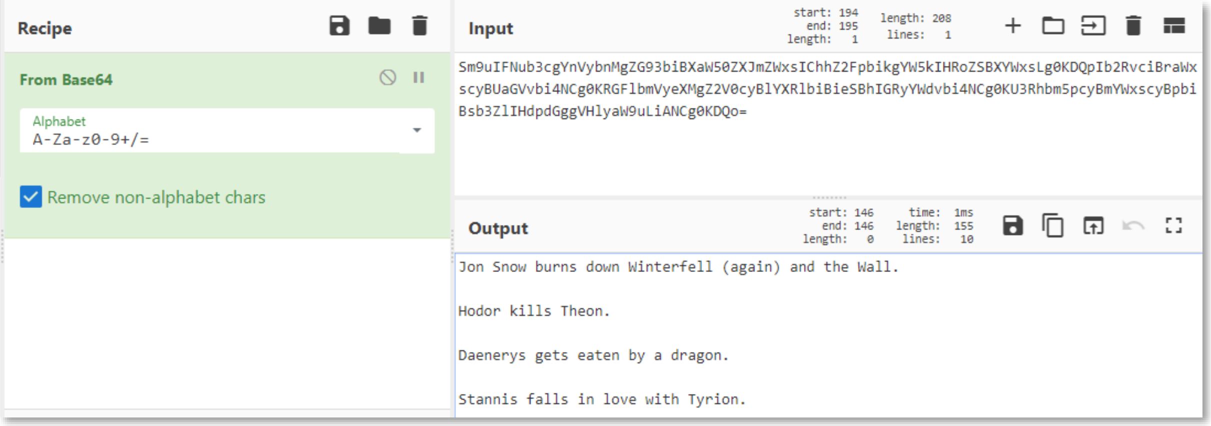
GoT Spoilers.docx

The files contained what looked like Base64 encoded text (Figure 8). When decoded using *CyberChef*, it revealed important plot events, or "spoilers", related to Game of Thrones as the filename suggested (Figure 9).



Sm9uIFNub3cgYnVybnMgZG93biBXaW
50ZXJmZWxsIChhZ2FpbikgYW5kIHRo
ZSBXYWxsLg0KDQpIb2RvciBraWxscy
BUaGVvbi4NCg0KRGFlbmVyeXMgZ2V
0cyBlYXRlbiBieSBhIGRyYWdvbi4NCg0
KU3Rhbm5pcyBmYWxscyBpbiBsb3ZlIH
dpdGggVHlyaW9uLiANCg0KDQo=

Figure 8



From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

```
start: 194    length: 208
  end: 195     lines:   1
length:  1
```

```
Sm9uIFNub3cgYnVybnMgZG93biBXaW50ZXJmZWxsIChhZ2FpbikgYW5kIHRoZSBXYWxsLg0KDQpIb2RvciBraWx
scyBUaGVvbi4NCg0KRGFlbmVyeXMgZ2V0cyBlYXRlbiBieSBhIGRyYWdvbi4NCg0KU3Rhbm5pcyBmYWxscyBpbi
Bsb3ZlIHdpdGggVHlyaW9uLiANCg0KDQo=
```

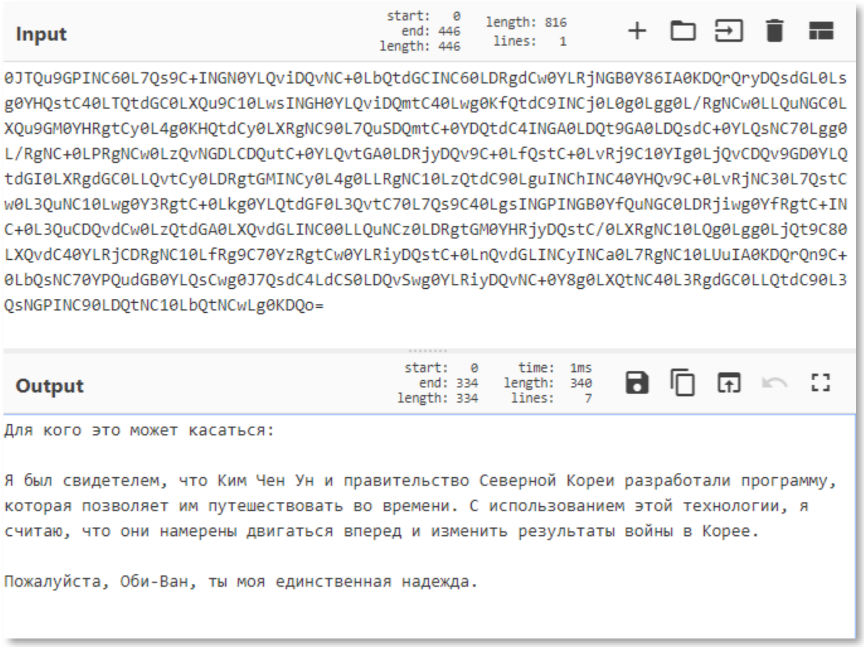
```
start: 146      time:  1ms
  end: 146    length:  155
length:  0     lines:   10
```

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

Hodor kills Theon.

Daenerys gets eaten by a dragon.

Stannis falls in love with Tyrion.
```

Figure 9

NorthKorea.docx

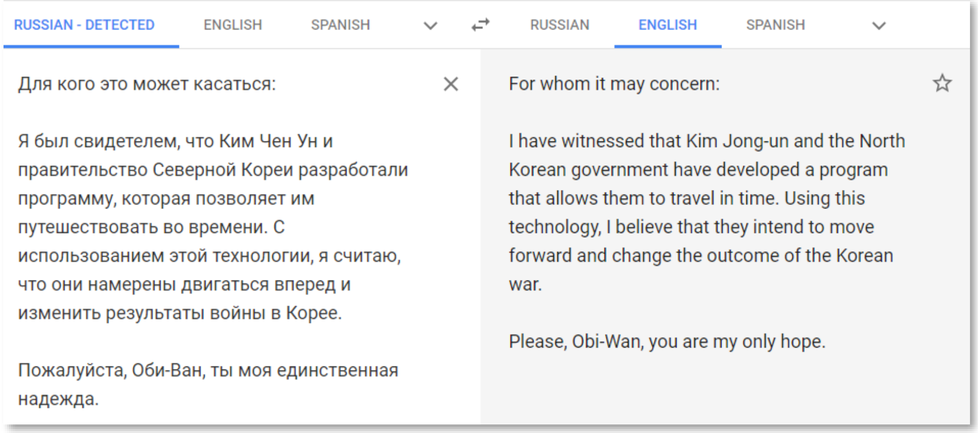
The document had Base64 encoded text inside which decoded into what looked like Russian language (Figure 10). Using an online translation tool, the text expressed concern about a North Korean time travel program (Figure 11).



```
0JTQu9GPINC60L7Qs9C+INGN0YLQviDQvNC+0LbQtdGCINC60LDRgdCw0YLRjNGB0Y86IA0KDQrQryDQsdGL0Ls
g0YHQstC40LTQtdGC0LXQu9C10LwsINGH0YLQviDQmtC40Lwg0KfQtdC9INCj0L0g0Lgg0L/RgNCw0LLQuNGC0L
XQu9GM0YHRgtCy0L4g0KHQtdCy0LXRgNC90L7QuSDQmtC+0YDQtdC4INGA0LDQt9GA0LDQsdC+0YLQsNC70Lgg0
L/RgNC+0LPRgNCw0LzQvNGDLCDQutC+0YLQvtGA0LDRjyDQv9C+0LfQstC+0LvRj9C10YIg0LjQvCDQv9GD0YLQ
tdGI0LXRgdGC0LLQvtCy0LDRgtGMINCy0L4g0LLRgNC10LzQtdC90LguINChINC40YHQv9C+0LvRjNC30L7QstC
w0L3QuNC10Lwg0Y3RgtC+0Lkg0YLQtdGF0L3QvtC70L7Qs9C40LgsINGPINGB0YfQuNGC0LDRjiwg0YfRgtC+IN
C+0L3QuCDQvdCw0LzQtdGA0LXQvdGLINC00LLQuNCz0LDRgtGM0YHRjyDQstC/0LXRgNC10LQg0Lgg0LjQt9C80
LXQvdC40YLRjCDRgNC10LfRg9C70YzRgtCw0YLRiyDQstC+0LnQvdGLINCyINCa0L7RgNC10LUuIA0KDQrQn9C+
0LbQsNC70YPQudGB0YLQsCwg0J7QsdC4LdCS0LDQvSwg0YLRiyDQvNC+0Y8g0LXQtNC40L3RgdGC0LLQtdC90L3
QsNGPINC90LDQtNC10LbQtNCwLg0KDQo=
```

```
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.
```

Figure 10



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.

For whom it may concern:

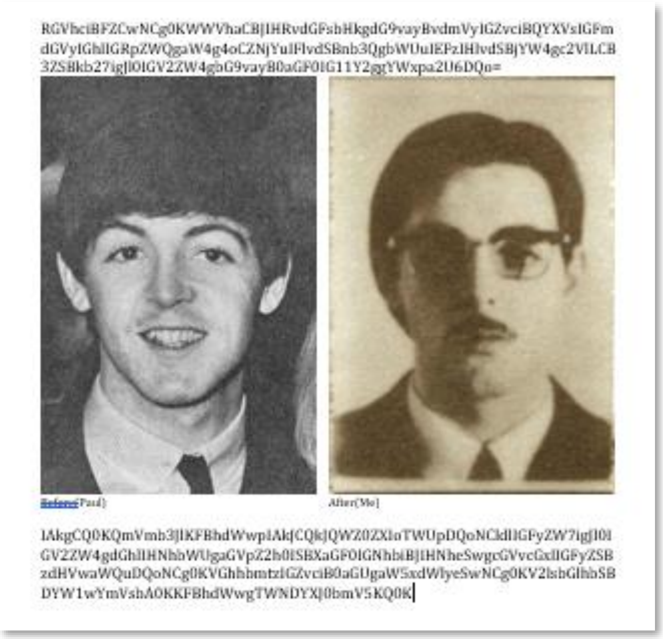
I have witnessed that Kim Jong-un and the North Korean government have developed a program that allows them to travel in time. Using this technology, I believe that they intend to move forward and change the outcome of the Korean war.

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

Figure 11

PiD.docx

The file contained Base64 encoded text and two photographs (Figure 12). When decoded, it was a message from a William Campbell, who has been impersonating Paul McCartney, to someone named Ed (Figures 13 & 14).



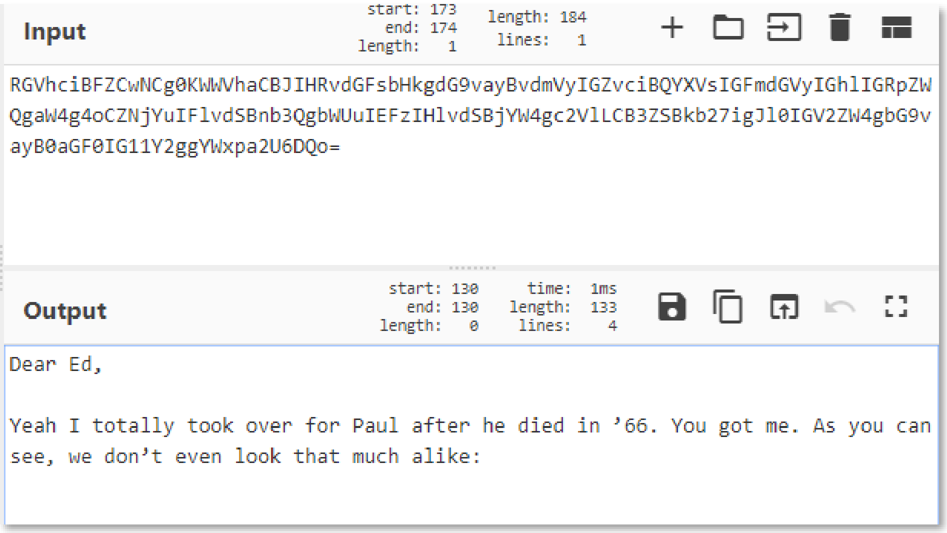
RGVhciBFZCwNCg0KWWVhaCBJIHRvdGFsbHkgdG9vayBvdmVyIGZvciBQYXVsIGFm
dGVyIGhlIGRpZWQgaW4g4oCZNjYuIFlvdSBnb3QgbWUuIEFzIHlvdSBjYW4gc2VlLCB
3ZSBkb27igJl0IGV2ZW4gbG9vayB0aGF0IG11Y2ggYWxpa2U6DQo=

Before(Paul)

After(Me)

IAkgCQ0KQmVmb3JlIKFBhdWwpIAkJCQkJQWZ0ZXIoTWUpDQoNCkdlIGFyZW7igJl0I
GV2ZW4gdGhlIHNhbWUgaGVpZ2h0ISBXaGF0IGNhbiBJIHNheSwgcGVvcGxlIGFyZSB
zdHVwaWQuDQoNCg0KVGhhbmtzIGZvciB0aGUgaW5xdWlyeSwNCg0KV2lsbGlhbSB
DYW1wYmVsbA0KKFBhdWwgTWNDYXJ0bmV5KQ0K

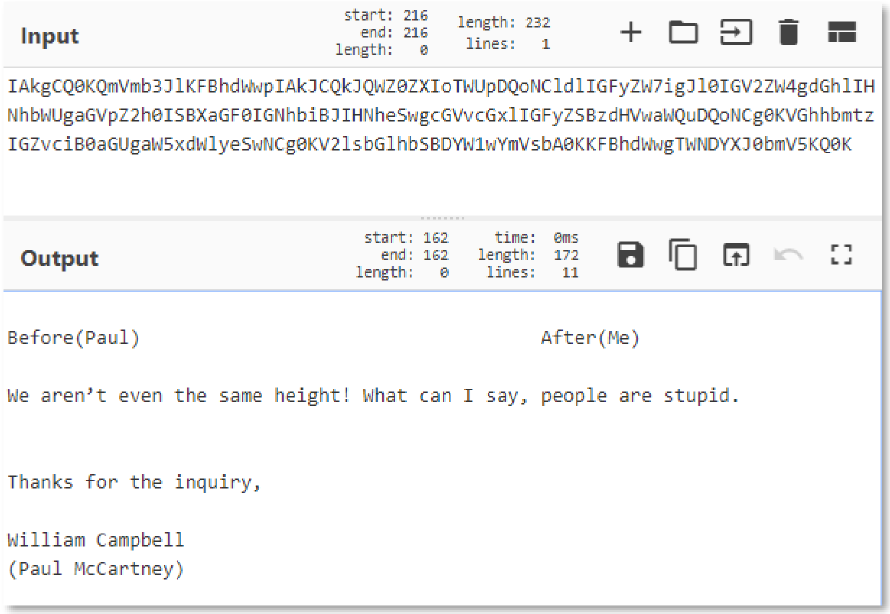
Figure 12



RGVhciBFZCwNCg0KWWVhaCBJIHRvdGFsbHkgdG9vayBvdmVyIGZvciBQYXVsIGFmdGVyIGhlIGRpZWQgaW4g4oCZNjYuIFlvdSBnb3QgbWUuIEFzIHlvdSBjYW4gc2VlLCB3ZSBkb27igJl0IGV2ZW4gbG9vayB0aGF0IG11Y2ggYWxpa2U6DQo=

```
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:
```

Figure 13



```
IAkgCQ0KQmVmb3JlKFBhdWwpIAkJCQkJQWZ0ZXIoTWUpDQoNCldlIGFyZW7igJl0IGV2ZW4gdGhlIH
NhbWUgaGVpZ2h0ISBXaGF0IGNhbiBJIHNheSwgcGVvcGxlIGFyZSBzdHVwaWQuDQoNCg0KVGhhbmtz
IGZvciB0aGUgaW5xdWlyeSwNCg0KV2lsbGlhbSBDYW1wYmVsbA0KKFBhdWwgTWNDYXJ0bmV5KQ0K
```

```

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)
```

Figure 14

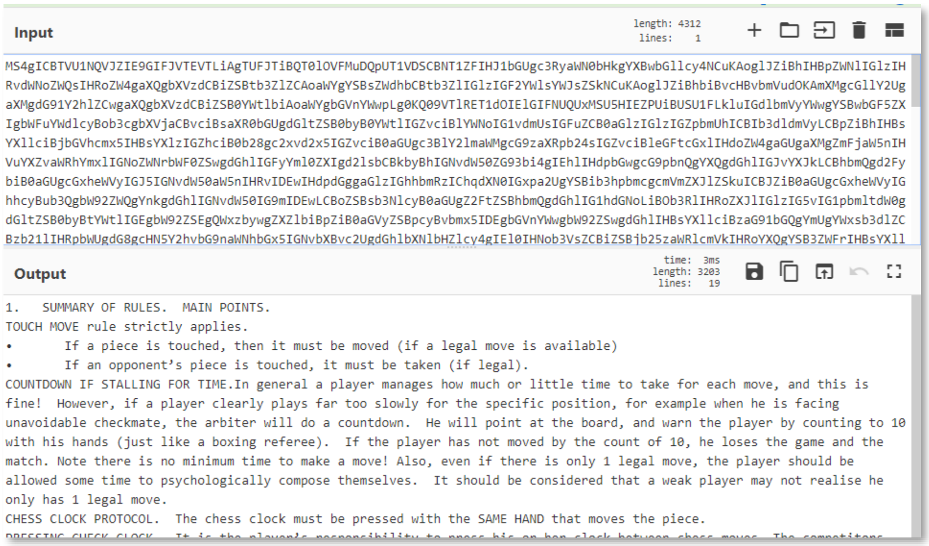
Chess Boxing

NK.jpg

Image of the North Korean flag; further analysis with *binwalk* did not reveal any hidden files. Even though the file was named like a JPG file, it was actually a PNG image, and because of this it could not be opened in the previously mentioned *SilentEye* tool.

Rules 1.docx – Rules 7.docx

Each document contained Base64 encoded text, and they decoded into what seem like the rules to “chess boxing” (Figure 15).



```
MS4gICBTVU1NQVJZIE9GIFJVTEVTLiAgTUFJTiBQT0lOVFMuDQpUT1VDSCBNT1ZFIHJ1bGUgc3RyaWN0bHkgYXBwbGllcy4NCuKAoglJZiBhIHBpZWNlIGlzIH
RvdWNoZWQsIHRoZW4gaXQgbXVzdCBiZSBtb3ZlZCAoaWYgYSBsZWdhbCBtb3ZlIGlzIGF2YWlsYWJsZSkNCuKAoglJZiBhbiBvcHBvbmVudOKAmXMgcGllY2Ug
aXMgdG91Y2hlZCwgaXQgbXVzdCBiZSB0YWtlbiAoaWYgbGVnYWwpLg0KQ09VTlRET1dOIElGIFNUQUxMSU5HIEZPUiBUSU1FLkluIGdlbmVyYWwgYSBwbGF5ZX
IgbWFuYWdlcyBob3cgbXVjaCBvciBsaXR0bGUgdGltZSB0byB0YWtlIGZvciBlYWNoIG1vdmUsIGFuZCB0aGlzIGlzIGZpbmUhICBIb3dldmVyLCBpZiBhIHBs
YXllciBjbGVhcmx5IHBsYXlzIGZhciB0b28gc2xvd2x5IGZvciB0aGUgc3BlY2lmaWMgcG9zaXRpb24sIGZvciBleGFtcGxlIHdoZW4gaGUgaXMgZmFjaW5nIH
VuYXZvaWRhYmxlIGNoZWNrbWF0ZSwgdGhlIGFyYml0ZXIgd2lsbCBkbyBhIGNvdW50ZG93bi4gIEhlIHdpbGwgcG9pbnQgYXQgdGhlIGJvYXJkLCBhbmQgd2Fy
biB0aGUgcGxheWVyIGJ5IGNvdW50aW5nIHRvIDEwIHdpdGggaGlzIGhhbmRzIChqdXN0IGxpa2UgYSBib3hpbmcgcmVmZXJlZSkuICBJZiB0aGUgcGxheWVyIG
hhcyBub3QgbW92ZWQgYnkgdGhlIGNvdW50IG9mIDEwLCBoZSBsb3NlcyB0aGUgZ2FtZSBhbmQgdGhlIG1hdGNoLiBOb3RlIHRoZXJlIGlzIG5vIG1pbmltdW0g
dGltZSB0byBtYWtlIGEgbW92ZSEgQWxzbywgZXZlbiBpZiB0aGVyZSBpcyBvbmx5IDEgbGVnYWwgbW92ZSwgdGhlIHBsYXllciBzaG91bGQgYmUgYWxsb3dlZC
Bzb21lIHRpbWUgdG8gcHN5Y2hvbG9naWNhbGx5IGNvbXBvc2UgdGhlbXNlbHZlcy4gIEl0IHNob3VsZCBiZSBjb25zaWRlcmVkIHRoYXQgYSB3ZWFrIHBsYXll
```

```
1.   SUMMARY OF RULES.  MAIN POINTS.
TOUCH MOVE rule strictly applies.
•       If a piece is touched, then it must be moved (if a legal move is available)
•       If an opponent’s piece is touched, it must be taken (if legal).
COUNTDOWN IF STALLING FOR TIME.In general a player manages how much or little time to take for each move, and this is
fine!  However, if a player clearly plays far too slowly for the specific position, for example when he is facing
unavoidable checkmate, the arbiter will do a countdown.  He will point at the board, and warn the player by counting to 10
with his hands (just like a boxing referee).  If the player has not moved by the count of 10, he loses the game and the
match. Note there is no minimum time to make a move! Also, even if there is only 1 legal move, the player should be
allowed some time to psychologically compose themselves.  It should be considered that a weak player may not realise he
only has 1 legal move.
CHESS CLOCK PROTOCOL.  The chess clock must be pressed with the SAME HAND that moves the piece.
PRESSING CHECK CLOCK.  It is the player’s responsibility to press his or her clock between chess moves. The competitors
```

Figure 15

Enter the WuTang

track6.docx

Base64 encoded text which decoded into a list of usernames who might be of interest to the suspected bribery investigation:

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

track10.docx

Base64 encoded text which decoded into the lyrics to the song "Protect Ya Neck" by Wu-Tang Clan.

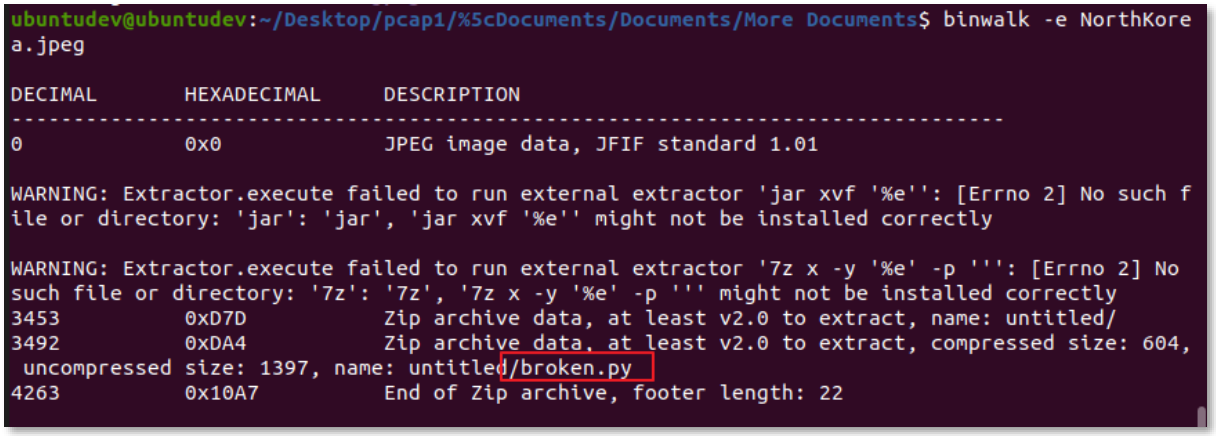
More Documents

BillOfRights.txt

Text document with a plain text transcription of the United States' Bill of Rights.

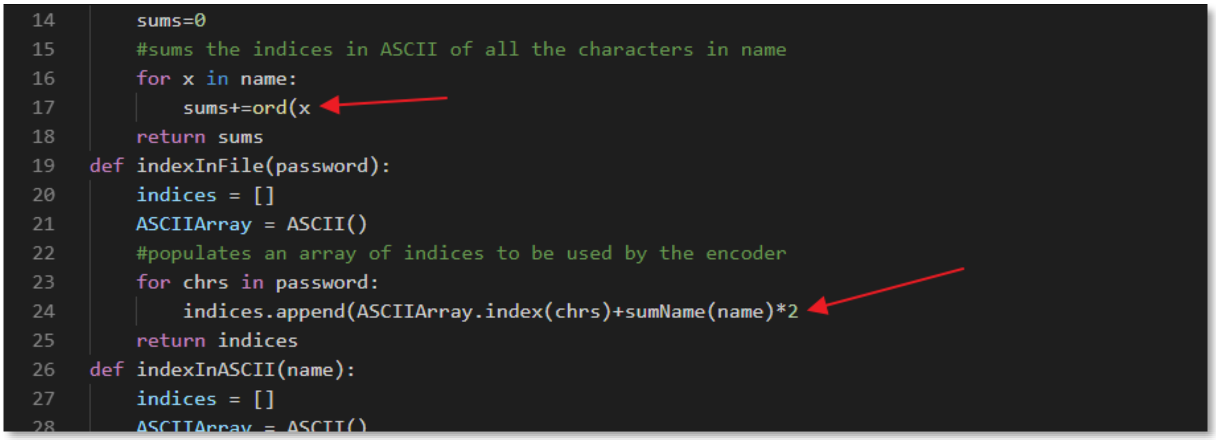
NorthKorea.jpeg

Image of the North Korean flag. When *binwalk* was used on the file, a python file *broken.py* was extracted (Figure 16). When it was run an error was displayed and on a closer examination, the script had several parentheses missing in different places (Figure 17). After fixing these, no errors were given but no output was shown either, and it seemed that the script was still incomplete. However, at this point the investigation was concluded as there were no further hints about how to fix the broken python script. The original image was imported into *SilentEye*, however the tool needed a password to decode any information. Several different passwords were tried, along with the default 'SilentEye' but the attempts were unsuccessful in recovering anything new.



```
ubuntudev@ubuntudev:~/Desktop/pcap1/%5cDocuments/Documents/More Documents$ binwalk -e NorthKore
a.jpeg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01

WARNING: Extractor.execute failed to run external extractor 'jar xvf '%e'': [Errno 2] No such f
ile or directory: 'jar': 'jar', 'jar xvf '%e'' might not be installed correctly

WARNING: Extractor.execute failed to run external extractor '7z x -y '%e' -p ''': [Errno 2] No
such file or directory: '7z': '7z', '7z x -y '%e' -p ''' might not be installed correctly
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: untitled/broken.py
4263          0x10A7          End of Zip archive, footer length: 22
```

Figure 16



```
    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()
    #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):
    indices = []
    ASCIIArray = ASCII()
```

Figure 17

REFERENCES CAPTURE 1.PCAP

No references

2 INVESTIGATION OF CAPTURE 2.PCAP

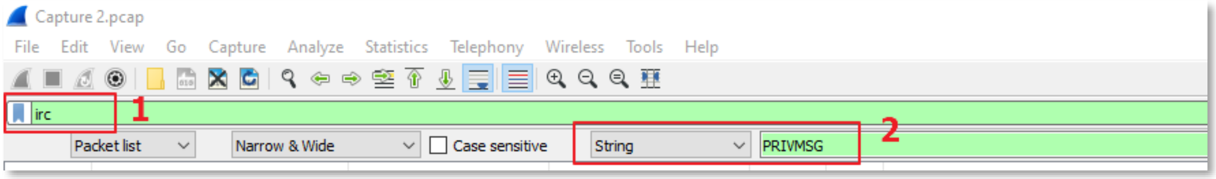
2.1 ABSTRACT

A PCAP file was provided that contained IRC messages encoded in several different ways, including Base64, Base32 and hexadecimal. After decoding the conversations, three officials (Razor1, Genius1 and Raekwon) showed intent to taking a bribe while two (Method, Killah) showed no interest. The messages also mentioned the possible locations of four of the five officials:

- Razor1: Pyongyang, North Korea
- Genius1: Caracas, Venezuela
- Raekwon: Russia/other Eastern European country
- Method: no indication of location
- Killah: Qatar

2.2 PROCEDURE

The PCAP file was searched for any IRC traffic using the filter 'irc' (Figure 18; 1) and to find each private message, the string PRIVMSG was used as a search term (Figure 18; 2):


Capture 2.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
1
2

irc

Packet list | Narrow & Wide | ☐ Case sensitive | String | PRIVMSG

Figure 18

In addition, the packet list section was configured to display the time for each packet so that each private message could be given a timestamp of when it was sent (Figure 19).

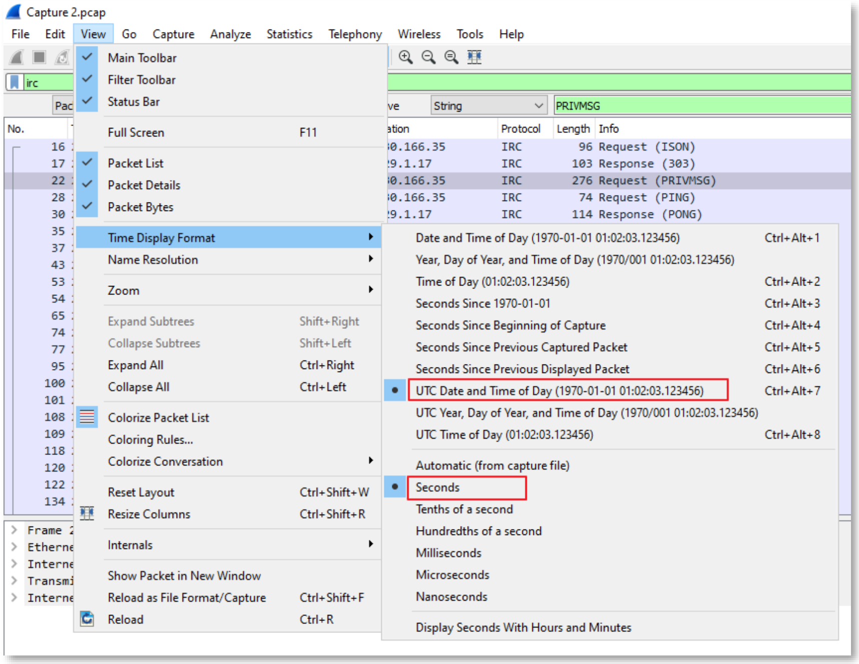
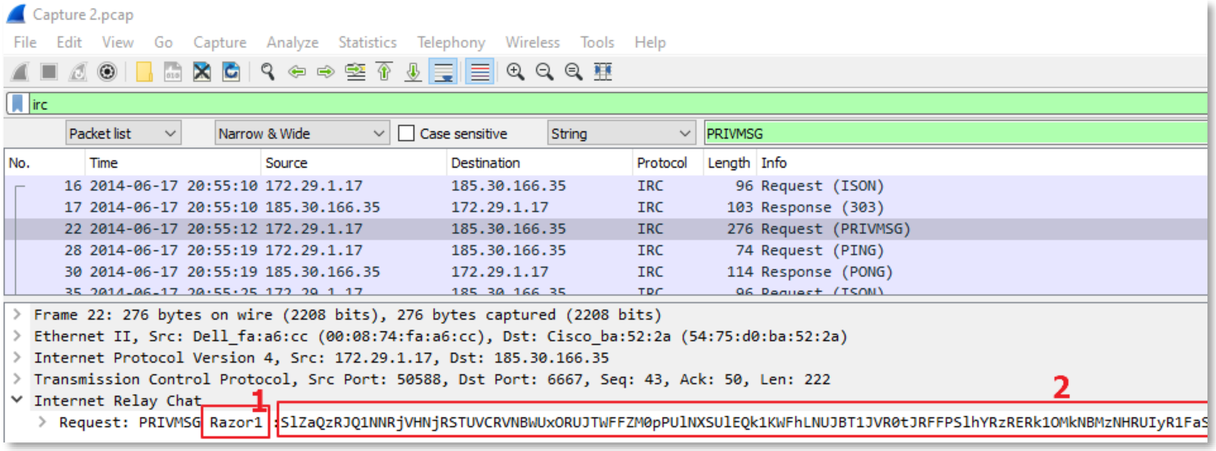

Capture 2.pcap
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help
irc
String
PRIVMSG
✓ Main Toolbar
✓ Filter Toolbar
✓ Status Bar
Full Screen F11
✓ Packet List
✓ Packet Details
✓ Packet Bytes
Time Display Format
Name Resolution
Zoom
Expand Subtrees Shift+Right
Collapse Subtrees Shift+Left
Expand All Ctrl+Right
Collapse All Ctrl+Left
Colorize Packet List
Coloring Rules...
Colorize Conversation
Reset Layout Ctrl+Shift+W
Resize Columns Ctrl+Shift+R
Internals
Show Packet in New Window
Reload as File Format/Capture Ctrl+Shift+F
Reload Ctrl+R
Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt+1
Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
Time of Day (01:02:03.123456) Ctrl+Alt+2
Seconds Since 1970-01-01 Ctrl+Alt+3
Seconds Since Beginning of Capture Ctrl+Alt+4
Seconds Since Previous Captured Packet Ctrl+Alt+5
Seconds Since Previous Displayed Packet Ctrl+Alt+6
● UTC Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt+7
UTC Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
UTC Time of Day (01:02:03.123456) Ctrl+Alt+8
Automatic (from capture file)
● Seconds
Tenths of a second
Hundredths of a second
Milliseconds
Microseconds
Nanoseconds
Display Seconds With Hours and Minutes

Figure 19

The content of each message could be viewed in the packet details section under Internet Relay Chat. The output shows the sender (Figure 20; 1) and the message (Figure 20; 2) which was encoded in several different ways:

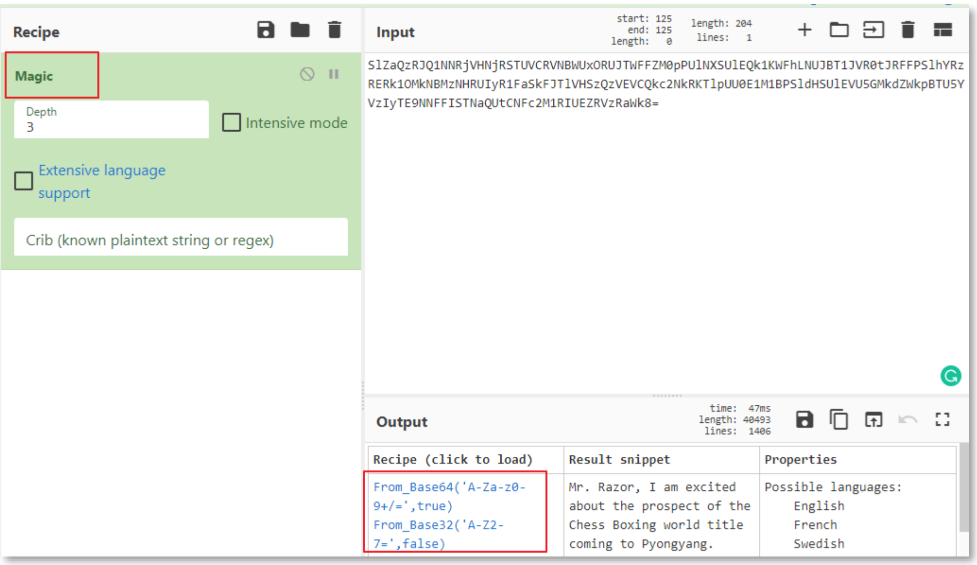


```
No.  Time                Source         Destination    Protocol Length Info
  16 2014-06-17 20:55:10 172.29.1.17    185.30.166.35  IRC        96 Request (ISON)
  17 2014-06-17 20:55:10 185.30.166.35  172.29.1.17    IRC       103 Response (303)
  22 2014-06-17 20:55:12 172.29.1.17    185.30.166.35  IRC       276 Request (PRIVMSG)
  28 2014-06-17 20:55:19 172.29.1.17    185.30.166.35  IRC        74 Request (PING)
  30 2014-06-17 20:55:19 185.30.166.35  172.29.1.17    IRC       114 Response (PONG)
  35 2014-06-17 20:55:25 172.29.1.17    185.30.166.35  IRC        96 Request (ISON)
```

```
> Frame 22: 276 bytes on wire (2208 bits), 276 bytes captured (2208 bits)
> Ethernet II, Src: Dell_fa:a6:cc (00:08:74:fa:a6:cc), Dst: Cisco_ba:52:2a (54:75:d0:ba:52:2a)
> Internet Protocol Version 4, Src: 172.29.1.17, Dst: 185.30.166.35
> Transmission Control Protocol, Src Port: 50588, Dst Port: 6667, Seq: 43, Ack: 50, Len: 222
v Internet Relay Chat
  > Request: PRIVMSG Razor1 :SlZaQzRJQ1NNRjVHNjRSTUVCRVNBWUxORUJTWFFZM0pPUlNXSUlEQk1KWFhLNUJBT1JVR0tJRFFPSlhYRzRERk1OMkNBMzNHRUIyR1FaS
```

Figure 20

To decode the message, *CyberChef* was used with its ‘magic’ feature which attempts to recognize any decoding (Figure 21). To view the whole message (Figure 22), the recognized “recipe” was chosen and the tool decoded the message (this extra step was done because the ‘magic’ feature only shows a snippet of the decoded message).



Magic

Depth
3

□ Intensive mode

□ Extensive language support

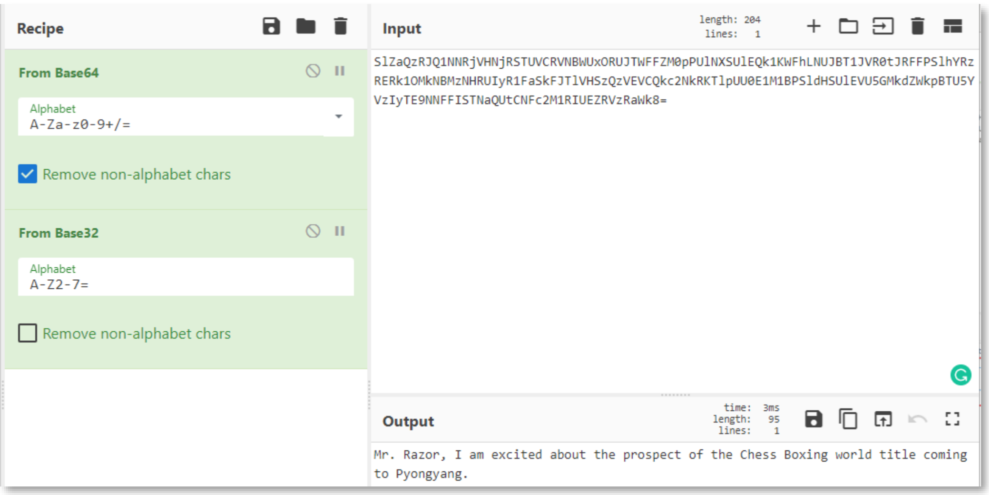
Crib (known plaintext string or regex)

```
SlZaQzRJQ1NNRjVHNjRSTUVCRVNBWUxORUJTWFFZM0pPUlNXSUlEQk1KWFhLNUJBT1JVR0tJRFFPSlhYRz
RERk1OMkNBMzNHRUIyR1FaSkFJTlVHSzQzVEVCQkc2NkRKTlpUU0E1M1BPSldHSUlEVU5GMkdZWkpBTU5Y
VzIyTE9NNFFISTNaQUtCNFc2M1RIUEZRVzRaWk8=
```

Recipe (click to load)	Result snippet	Properties

From_Base64('A-Za-z0-9+/=',true)
From_Base32('A-Z2-7=',false)
	Mr. Razor, I am excited about the prospect of the Chess Boxing world title coming to Pyongyang.	Possible languages:
		English
		French
		Swedish

Figure 21



From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

From Base32

Alphabet
A-Z2-7=

□ Remove non-alphabet chars

```
SlZaQzRJQ1NNRjVHNjRSTUVCRVNBWUxORUJTWFFZM0pPUlNXSUlEQk1KWFhLNUJBT1JVR0tJRFFPSlhYRz
RERk1OMkNBMzNHRUIyR1FaSkFJTlVHSzQzVEVCQkc2NkRKTlpUU0E1M1BPSldHSUlEVU5GMkdZWkpBTU5Y
VzIyTE9NNFFISTNaQUtCNFc2M1RIUEZRVzRaWk8=
```

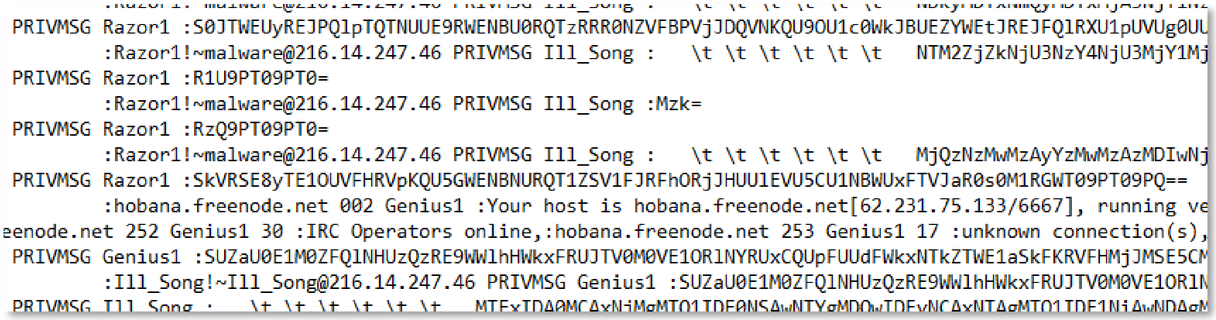
```
Mr. Razor, I am excited about the prospect of the Chess Boxing world title coming
to Pyongyang.
```

Figure 22

To quickly filter out all other packets except the ones with a message, *tshark* was used with the following command:

C:\Program Files\Wireshark> .\tshark.exe -nr 'C:\Path\To\PCAP’ -Y 'frame contains "PRIVMSG" and irc.response.trailer or irc.request.trailer' -T fields -e frame.number -e _ws.col.UTC -e ip.src -e ip.dst -e irc.request -e irc.response > 'C:\Path\For\Output

The output file was then examined (Figure 23), and each message was decoded and transcribed.

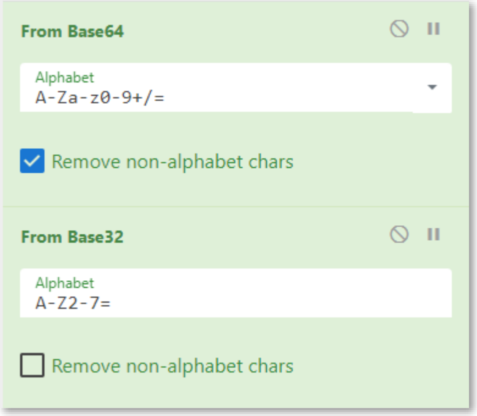


```
        :Razor1!~malware@216.14.247.46 PRIVMSG Ill_Song :   \t \t \t \t \t \t
 PRIVMSG Razor1 :S0JTWEUyREJPQ1pTQTNUUE9RWENBU0RQTzRRR0NZVFBPVjJDQVNKQU9OU1c0WkJBUEZYWEtJREJFQ1RXU1pUVUg0UU
        :Razor1!~malware@216.14.247.46 PRIVMSG Ill_Song :   \t \t \t \t \t \t   NTM2ZjZkNjU3NzY4NjU3MjY1Mj
 PRIVMSG Razor1 :R1U9PT09PT0=
        :Razor1!~malware@216.14.247.46 PRIVMSG Ill_Song :Mzk=
 PRIVMSG Razor1 :RzQ9PT09PT0=
        :Razor1!~malware@216.14.247.46 PRIVMSG Ill_Song :   \t \t \t \t \t \t   MjQzNzMwMzAyYzMwMzAzMDIwNj
 PRIVMSG Razor1 :SkVRSE8yTE1OUVFHRVpKQU5GWENBNURQT1ZSV1FJRFhORjJHUU1EVU5CU1NBWUxFTVJaR0s0M1RGWT09PT09PQ==
        :hobana.freenode.net 002 Genius1 :Your host is hobana.freenode.net[62.231.75.133/6667], running ve
eenode.net 252 Genius1 30 :IRC Operators online,:hobana.freenode.net 253 Genius1 17 :unknown connection(s),
 PRIVMSG Genius1 :SUZaU0E1M0ZFQ1NHUzQzRE9WWlhHWkxFRUJTV0M0VE1OR1NYRUxCQUpFUUdFWkxNTkZTWE1aSkFKRVFHMjJMSE5CM
        :Ill_Song!~Ill_Song@216.14.247.46 PRIVMSG Genius1 :SUZaU0E1M0ZFQ1NHUzQzRE9WWlhHWkxFRUJTV0M0VE1OR1N
 PRIVMSG Ill_Song :   \t \t \t \t \t \t   MTExIDA0MCAxNjMgMTQ1IDE0NSAwNTYgMDQwIDExNCAxNTAgMTQ1IDE1NiAwNDAgM
```

Figure 23

Decoding recipes:

The recipes needed to decode the messages were (can be confirmed by using the 'magic' feature):



From Base64

Alphabet
A-Za-z0-9+/=

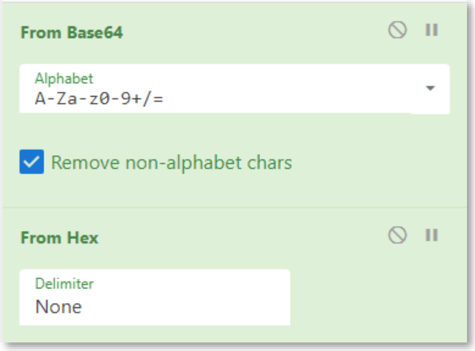
☑ Remove non-alphabet chars

From Base32

Alphabet
A-Z2-7=

☐ Remove non-alphabet chars

Figure 24 From Base64 -> From Base32 = plaintext.



From Base64

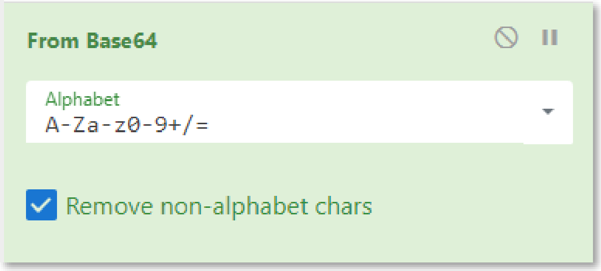
Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

From Hex

Delimiter
None

Figure 25 From Base64 -> From Hex = plaintext.

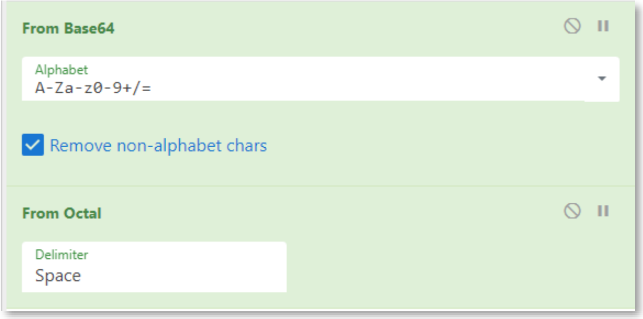


From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

Figure 26 From Base64 = plaintext.



From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

From Octal

Delimiter
Space

Figure 27 From Base64 -> From Octal = plaintext.

Transcript:

Message #1

2014-06-17 20:55:12

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Razor1:

Mr. Razor, I am excited about the prospect of the Chess Boxing world title coming to Pyongyang.

Message #2

2014-06-17 20:55:31

Decoding recipe: From Base64 -> From Hex = plaintext

Razor1 -> Ill_Song:

Well the decision is not final yet.

Message #3

2014-06-17 20:56:07

Decoding recipe: From Base64 -> From Hex = plaintext

Razor1 -> Ill_Song:

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.

Message #4

2014-06-17 20:56:25

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Razor1:

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.

Message #5

2014-06-17 20:56:49

Decoding recipe: From Base64 -> From Hex = plaintext

Razor1 -> Ill_Song:

Somewhere expensive, I hope.

Message #6

2014-06-17 20:57:03

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Razor1:

5

Message #7

2014-06-17 20:57:26

Decoding recipe: From Base64 = plaintext

Razor1-> Ill_Song:

39

Message #8

2014-06-17 20:58:03

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Razor1:

7

Message #9

2014-06-17 20:58:45

Decoding recipe: From Base64 -> From Hex = plaintext

Razor1 -> Ill_Song:

$700,000 it is. Where can I meet you?

Message #10

2014-06-17 20:59:13

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Razor1:

I will be in touch with the address.

Message #11

2014-06-17 21:00:57

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Genius1:

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

Message #12

2014-06-17 21:01:31

Decoding recipe: From Base64 -> From Octal = plaintext

Genius1 -> Ill_Song:

I see. Then we must meet, and I will see the validity of this claim.

Message #13

2014-06-17 21:02:12

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Genius1:

I can be in c9fa5b8cb3b197ae5ce4baf8415a375b within the week.

- ➔ MD5 hash decrypts to: ***Caracas***
- ➔ Full message: ***I can be in Caracas within the week.***

Message #14

2014-06-17 21:02:50

Decoding recipe: From Base64 -> From Octal = plaintext

Genius1 -> Ill_Song:

No. Not here. Can I not go to you?

Message #15

2014-06-17 21:03:38

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Genius1:

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.

Message #16

2014-06-17 21:04:33

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Method:

Mr. Method, I am excited about the prospect of the Chess Boxing world title coming to Pyongyang.

Message #17

2014-06-17 21:04:52

Decoding recipe: From Base64 -> From Hex = plaintext

Method -> Ill_Song:

I am not sure who you are, but I have an idea. Either way, I am not interested.

Message #18

2014-06-17 21:05:24

Decoding recipe: From Base64 -> Remove first line (salt?) -> From Base32 = plaintext

Ill_Song -> Method:

I am just hopeful. It would mean so much to have the Title here. Please consider it.

Note: After decoding the message first with Base64, the result was two separate strings. By decoding again with Base64 at this point made some of the message comprehensible but the rest was not. By removing the first string after the initial decoding and then decoding with Base32, the whole message was displayed correctly.

Message #19

2014-06-17 21:05:41

Decoding recipe: From Base64 -> From Hex = plaintext

Method -> Ill_Song:

Do not speak to me again.

Message #20

2014-06-17 21:06:19

Decoding recipe: From Base32 -> From Base32 = plaintext

Ill_Song -> Killah:

How is the weather in Qatar, Mr. Killah?

Message #21

2014-06-17 21:06:41

Decoding recipe: From Base64 -> From Octal = plaintext

Killah -> Ill_Song:

Hot, as always. Who is this?

Message #22

2014-06-17 21:07:01

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Killah:

I am a fan of Chess Boxing. I would love to see the Title held in Korea.

Message #23

2014-06-17 21:07:17

Decoding recipe: From Base64 -> From Octal = plaintext

Killah -> Ill_Song:

We will have to see how the bid turns out.

Message #24

2014-06-17 21:07:34

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Killah:

Is there anything that I could do to help make your decision easier?

Message #25

2014-06-17 21:08:04

Decoding recipe: From Base64 -> From Octal

Killah -> Ill_Song:

No! The great nation of Qatar would never be swayed so easily.

Message #26

2014-06-17 21:08:30

Decoding recipe: From Base64 -> From Octal = plaintext

Killah -> Ill_Song:

Nor would I. We do not take kindly to this pathetic notion of bribery.

Message #27

2014-06-17 21:09:46

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Raekwon:

Mr. Raekwon, have you spoken with Mr. Razor?

Message #28

2014-06-17 21:10:04

Decoding recipe: From Base64 -> From Hex = plaintext

Raekwon -> Ill_Song:

I have, but I won.t be bought so easily.

Message #29

2014-06-17 21:10:30

Decoding recipe: From Base64 -> From Base32

Ill_Song -> Raekwon:

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.

Message #30

2014-06-17 21:10:48

Decoding recipe: From Base64 -> From Hex = plaintext

Raekwon -> Ill_Song:

I would need at least 20 million Rubles.

Message #31

2014-06-17 21:11:46

Decoding recipe: From Base64 -> From Base32 = plaintext

Ill_Song -> Raekwon:

Consider it done. I will send you the information for the drop-off point soon.

Analysis:

Based on the language used in a message, the following officials show intent to taking a bribe or to meet Ill_Song:

Official: Razor1

Location: Pyongyang, North Korea (message #3)

Agrees to bribe: message #9

Official: Genius1

Location: Caracas, Venezuela (messages #13 and #14)

Agrees to meet: message #12

Official: Raekwon

Location: Russia or other related Eastern European countries who use rubles as their currency (message #30)

Agrees to bribe: message #30

The following officials did not agree to meet with Ill_Song:

Official: Method

Location: No indication; might be in the malformed message (message #18)

Turns down Ill_Song: messages #17 and #19

Official: Killah

Location: Qatar (messages #20 and #21)

Turns down Ill_Song: messages #25 and #26

REFERENCES CAPTURE 2.PCAP

Wireshark (no date) *Display Filter Reference: Internet Relay Chat*. Available at: https://www.wireshark.org/docs/dfref/i/irc.html (Accessed: 7 December 2019).

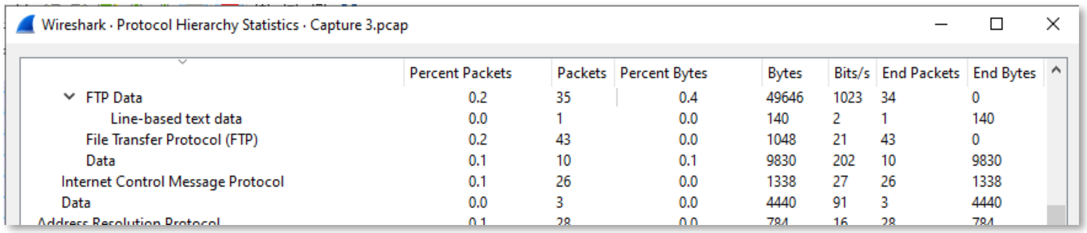
3 INVESTIGATION OF CAPTURE 3.PCAP

3.1 ABSTRACT

A PCAP file with FTP traffic between a suspected corrupt official and another malicious user (Ill-Song) was examined for possible file transfers. By exporting the raw hexadecimal bytes from the FTP packets, two ZIP files (sandofwhich.zip and ojd34.zip) with 20 jpg images, each named after an English word part of an Edward Snowden quote, were recovered. By further searching the PCAP for more ZIP files, an email message was found, and by extracting the hexadecimal bytes as before, three more ZIP files (34jdsioj.zip, breaking_bad_season_6.zip and canc3l.zip) with more similarly named images were carved. By using the images in the order of the quote, an image of a chess board was assembled which seemed to be the bribe the official received.

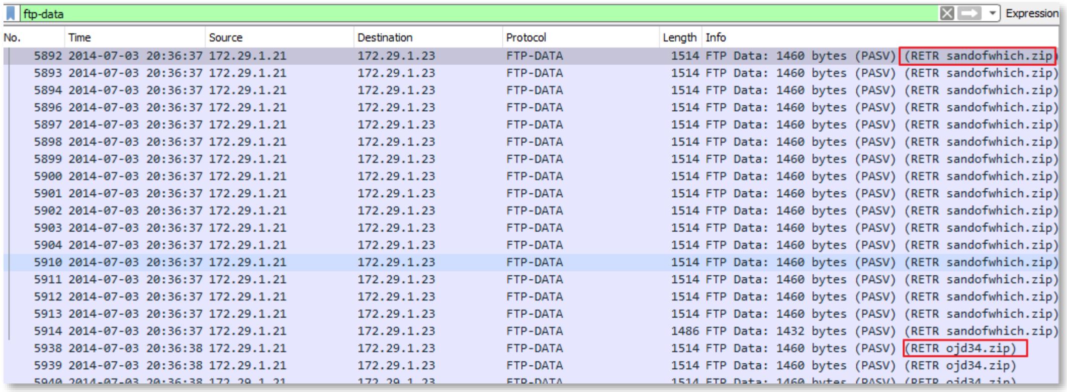
3.2 PROCEDURE

To start the investigation on the PCAP, the protocol hierarchy statistics were viewed (Statistics -> Protocol Hierarchy). The task was to investigate FTP traffic (Figure 28), it used as a filter to decrease the number of packets that had to be analysed. The matching packets suggested there were two ZIP files, sandofwhich.zip and ojd34.zip (Figure 29).


Wireshark · Protocol Hierarchy Statistics · Capture 3.pcap

	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
FTP Data	0.2	35	0.4	49646	1023	34	0
Line-based text data	0.0	1	0.0	140	2	1	140
File Transfer Protocol (FTP)	0.2	43	0.0	1048	21	43	0
Data	0.1	10	0.1	9830	202	10	9830
Internet Control Message Protocol	0.1	26	0.0	1338	27	26	1338
Data	0.0	3	0.0	4440	91	3	4440
Address Resolution Protocol	0.1	28	0.0	784	16	28	784

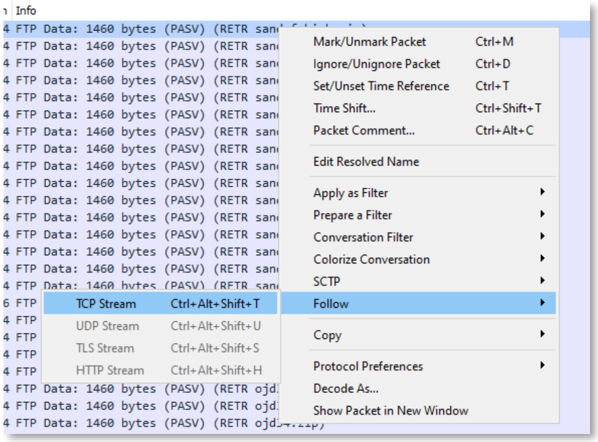
Figure 28


ftp-data

No.	Time	Source	Destination	Protocol	Length	Info
5892	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5893	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5894	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5896	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5897	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5898	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5899	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5900	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5901	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5902	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5903	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5904	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5910	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5911	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5912	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5913	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR sandofwhich.zip)
5914	2014-07-03 20:36:37	172.29.1.21	172.29.1.23	FTP-DATA	1486	FTP Data: 1432 bytes (PASV) (RETR sandofwhich.zip)
5938	2014-07-03 20:36:38	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
5939	2014-07-03 20:36:38	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)
5940	2014-07-03 20:36:38	172.29.1.21	172.29.1.23	FTP-DATA	1514	FTP Data: 1460 bytes (PASV) (RETR ojd34.zip)

Figure 29

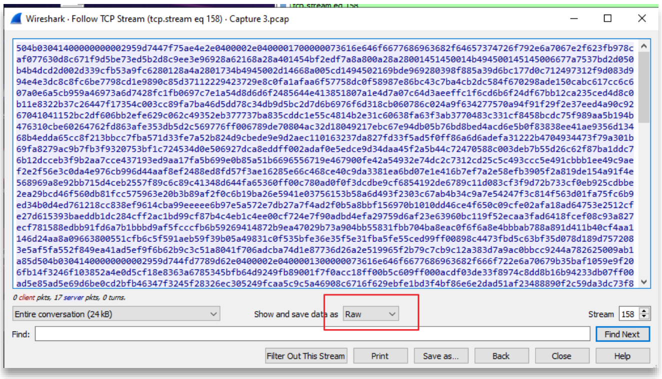
To try and carve out the first ZIP file (sandofwhich.zip) the first packet was chosen, and its TCP stream followed (Figure 30). The data was saved as raw hex (Figure 31) and then this file was examined with a hex editor (Figure 32). The beginning bytes, so called magic bytes, matched those of a ZIP file so nothing had to be edited out as sometimes there are unrelated header bytes when carving a file from Wireshark.



```
n Info
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
4 FTP Data: 1460 bytes (PASV) (RETR san
6 FTP
4 FTP
4 FTP
4 FTP
4 FTP
4 FTP Data: 1460 bytes (PASV) (RETR ojd
4 FTP Data: 1460 bytes (PASV) (RETR ojd
4 FTP Data: 1460 bytes (PASV) (RETR ojd
```

Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
TLS Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H

Figure 30



Wireshark · Follow TCP Stream (tcp.stream eq 158) · Capture 3.pcap

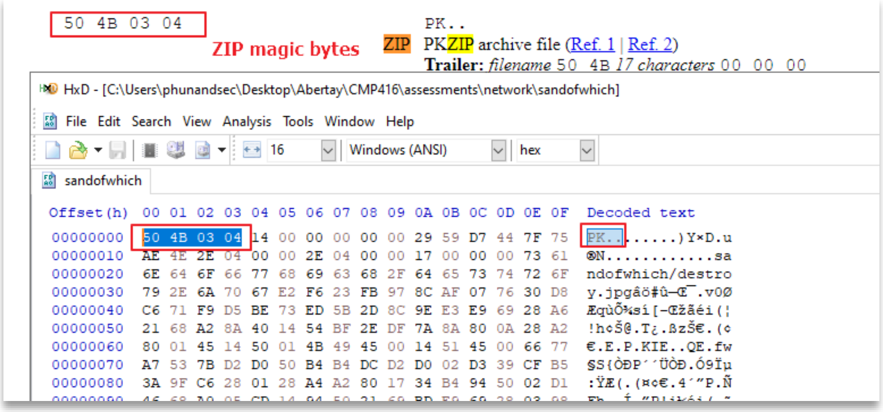
```
504b0304140000000000002959d7447f75ae4e2e0400002e0400001700000073616e646f6677686963682f64657374726f792e6a7067e2f623fb978c
af077630d8c671f9d5be73ed5b2d8c9ee3e96928a62168a28a401454bf2edf7a8a800a28a28001451450014b494500145145006677a7537bd2d050
b4b4dcd2d002d339cfb53a9fc6280128a4a2801734b4945002d14668a005cd1494502169bde969280398f885a39d6bc177d0c712497312f9d083d9
94e4e3dc8c8fc6be7798cd1e9890c85d37112229423729e8c0fa1afaa6f57758dc0f58987e86bc43c7ba4cb2dc584f670298ade150cabc617cc6c6
07a0e6a5cb959a46973a6d7428fc1fb0697c7e1a54d8d6d6f2485644e413851807a1e4d7a07c64d3aeeffc1f6cd6b6f24df67bb12ca235ced4d8c0
b11e8322b37c26447f17354c003cc89fa7ba46d5dd78c34db9d5bc2d7d6b6976f6d318cb060786c024a9f634277570a94f91f29f2e37eed4a90c92
67041041152bc2df606bb2efe629c062c49352eb377737ba835cddc1e55c4814b2e31c60638fa63f3ab3770483c331cf8458bcdc75f989aa5b194b
476310cbe60264762fd863afe353db5d2c569776ff006789de70804ac32d18049217ebc67e94db05b76bd8bed4acd6e5b0f83838ee41ae9356d134
68b4edda65cc8f213bbcc7fba571d33fe7a52b824d9cbede9e9d2aec110163237da827fd33f5ad5f0ff86a6d6adefa31222b4704934473f79a301b
69fa8279ac9b7fb3f9320753bf1c724534d0e506927dca8eddff002adaf0e5edce9d34daa45f2a5b44c72470588c003deb7b55d26c62f87ba1ddc7
6b12dcceb3f9b2aa7cce437193ed9aa17fa5b699e0b85a51b6696556719e467900fe42a54932e74dc2c7312cd25c5c493ccc5e491cbbb1ee49c9ae
f2e2f56e3c0da4e976cb996d44aaf8ef2488ed8fd57f3ae16285e66c468ce40c9da3381ea6bd07e1e416b7ef7a2e58efb3905f2a819de154a91f4e
568969a8e92bb715d4ceb2557f89c6c89c41348d644fa65360ff00c780ad0f0f3dcdbe9cf6854192de6789c11d083cf3f9d72b733cf0eb925cdbbe
2ea29bcd46f560db81fcc575963e20b3b89af2f0c6b19ba26e5941e03756153b58a6d493f2303c67ab4b34c9a7e54247f3c814f563d01fa75fc6b9
ed34b0d4ed761218cc838ef9614cba99eeeee6b97e5a572e7db27a7f4ad2f0b5a8bbf156970b1010dd46ce4f650c09cfe02afa18ad64753e2512cf
e27d615393baeddb1dc284cff2ac1bd99cf87b4c4eb1c4ee00cf724e7f90adbd4efa29759d6af23e63960bc119f52ecaa3fad6418fcef08c93a827
ecf781588edbb91fd6a7b1bbbd9af5fcccfb6b59269414872b9ea47029b73a904bb55831fbb704ba8eac0f6f6a8e4bbbab788a891d411b40cf4aa1
146d24aa8a09663800551cfb6c5f591aeb59f39b05a49831c0f535bfe36e35f5e31fba5fe55ced99ff00898c4473fbd5c63bf35d078d189d757208
3e5af5fa552f849ea41ad5ef9f6b62b9c3c51a8041f706adcba74d1e87736d26a2e519965f2b79c7cb9c12a383d7a9ac0bbcc9244a782625009ab1
a85d504b0304140000000000002959d744fd7789d62e0400002e0400001300000073616e646f6677686963682f666f722e6a70679b35baf1059e9f20
6fb14f3246f103852a4e0d5cf18e8363a6785345bfb64d9249fb89001f7f0acc18ff00b5c609ff000acdf03de33f8974c8dd8b16b94233db07ff00
ad5e85ad5e69d6be0cd2bfb46347f3245f28326ec305249fcaa5c9c5a46908c6716f629ebfe1bd3f4bf86e6e2dad51af23488890f2c59da3dc73f8
```

0 client pkts, 17 server pkts, 0 turns.

Entire conversation (24 kB) Show and save data as Raw Stream 158

Find:

Figure 31



50 4B 03 04
ZIP magic bytes
ZIP
PK..
PKZIP archive file (Ref. 1 | Ref. 2)
Trailer: filename 50 4B 17 characters 00 00 00

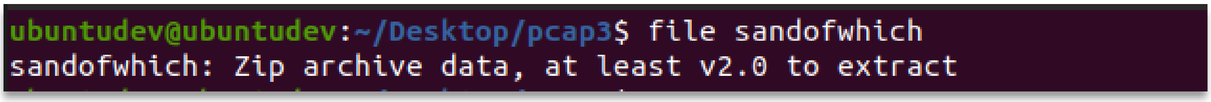
HxD - [C:\Users\phunandsec\Desktop\Abertay\CMP416\assessments\network\sandofwhich]

File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
sandofwhich

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  50 4B 03 04 14 00 00 00 00 00 29 59 D7 44 7F 75  PK......)Y×D.u
00000010  AE 4E 2E 04 00 00 2E 04 00 00 17 00 00 00 73 61  ®N............sa
00000020  6E 64 6F 66 77 68 69 63 68 2F 64 65 73 74 72 6F  ndofwhich/destro
00000030  79 2E 6A 70 67 E2 F6 23 FB 97 8C AF 07 76 30 D8  y.jpgâö#û—Œ¯.v0Ø
00000040  C6 71 F9 D5 BE 73 ED 5B 2D 8C 9E E3 E9 69 28 A6  ÆqùÕ¾sí[-Œžãéi(¦
00000050  21 68 A2 8A 40 14 54 BF 2E DF 7A 8A 80 0A 28 A2  !h¢Š@.T¿.ßzŠ€.(¢
00000060  80 01 45 14 50 01 4B 49 45 00 14 51 45 00 66 77  €.E.P.KIE..QE.fw
00000070  A7 53 7B D2 D0 50 B4 B4 DC D2 D0 02 D3 39 CF B5  §S{ÒÐP´´ÜÒÐ.Ó9Ïµ
00000080  3A 9F C6 28 01 28 A4 A2 80 17 34 B4 94 50 02 D1  :ŸÆ(.(¤¢€.4´”P.Ñ
```

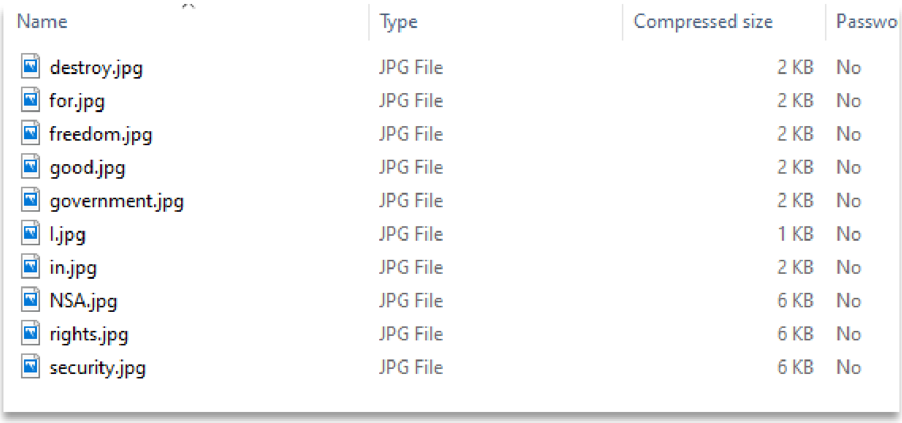
Figure 32

The file was then copied onto an Ubuntu virtual machine and the command *file* was used to further verify that it was a ZIP archive (Figure 33). The file could then simply be renamed .zip and opened. The archive seemed to contain a set of 10 JPG images (Figure 34) but when they were examined a file format error was displayed (Figure 35).



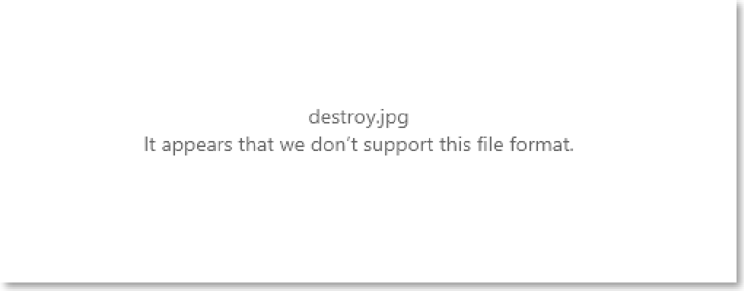
```
ubuntudev@ubuntudev:~/Desktop/pcap3$ file sandofwhich
sandofwhich: Zip archive data, at least v2.0 to extract
```

Figure 33



Name	Type	Compressed size	Passwo
destroy.jpg	JPG File	2 KB	No
for.jpg	JPG File	2 KB	No
freedom.jpg	JPG File	2 KB	No
good.jpg	JPG File	2 KB	No
government.jpg	JPG File	2 KB	No
I.jpg	JPG File	1 KB	No
in.jpg	JPG File	2 KB	No
NSA.jpg	JPG File	6 KB	No
rights.jpg	JPG File	6 KB	No
security.jpg	JPG File	6 KB	No

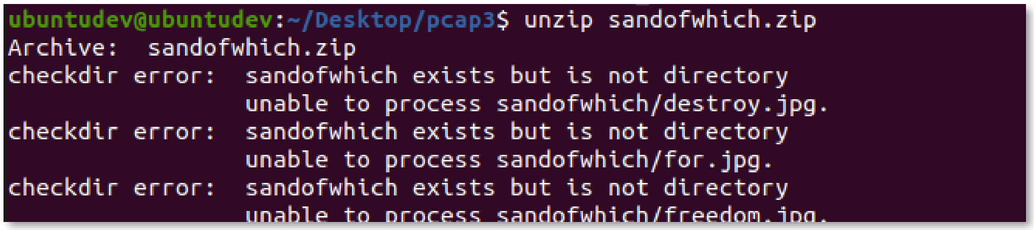
Figure 34



destroy.jpg
It appears that we don't support this file format.

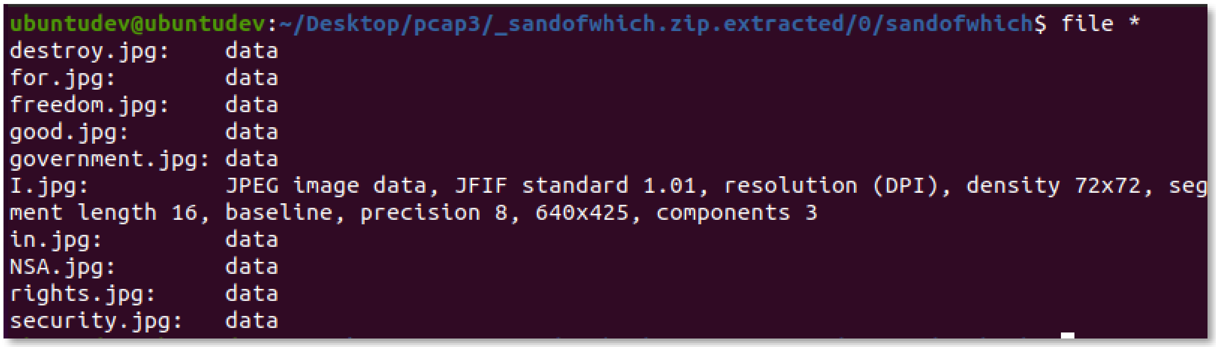
Figure 35

The files were then attempted to be extracted in Ubuntu, but an error was displayed (Figure 36). The files were then attempted to be carved out by using *binwalk* and this time the files were accessible but when *file* was run against them, only one seemed to be an image (Figure 37).



```
ubuntudev@ubuntudev:~/Desktop/pcap3$ unzip sandofwhich.zip
Archive:  sandofwhich.zip
checkdir error:  sandofwhich exists but is not directory
                 unable to process sandofwhich/destroy.jpg.
checkdir error:  sandofwhich exists but is not directory
                 unable to process sandofwhich/for.jpg.
checkdir error:  sandofwhich exists but is not directory
                 unable to process sandofwhich/freedom.jpg.
```

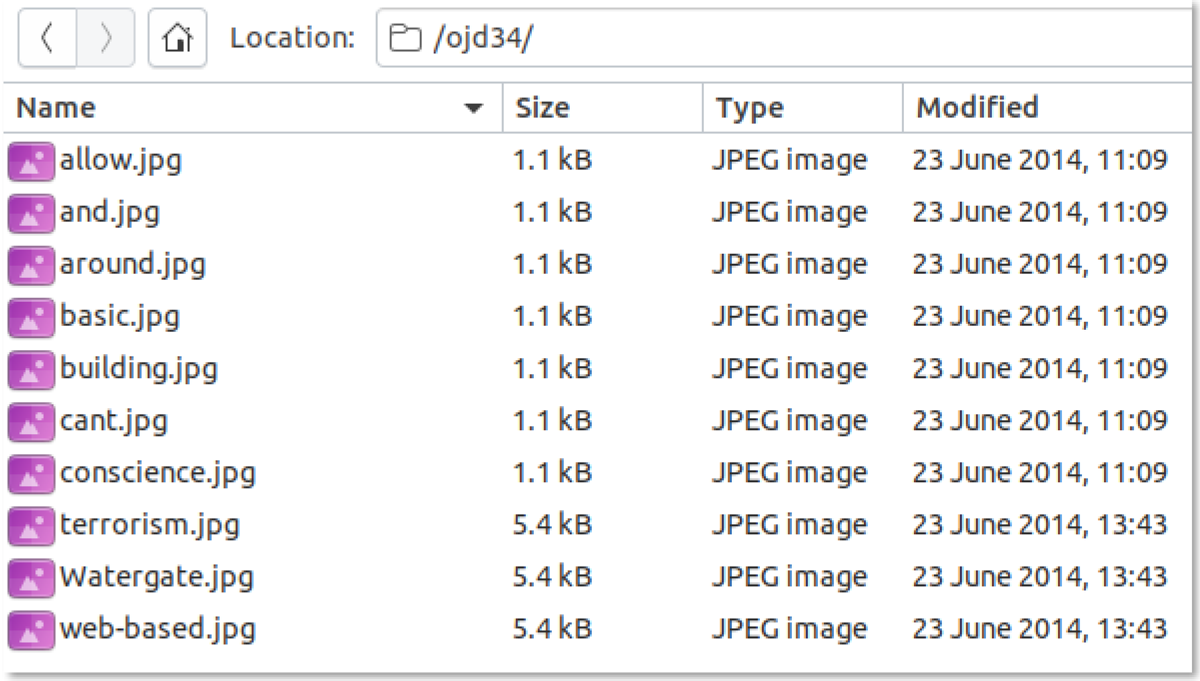
Figure 36



```
ubuntudev@ubuntudev:~/Desktop/pcap3/_sandofwhich.zip.extracted/0/sandofwhich$ file *
destroy.jpg:    data
for.jpg:        data
freedom.jpg:    data
good.jpg:       data
government.jpg: data
I.jpg:          JPEG image data, JFIF standard 1.01, resolution (DPI), density 72x72, seg
ment length 16, baseline, precision 8, 640x425, components 3
in.jpg:         data
NSA.jpg:        data
rights.jpg:     data
security.jpg:   data
```

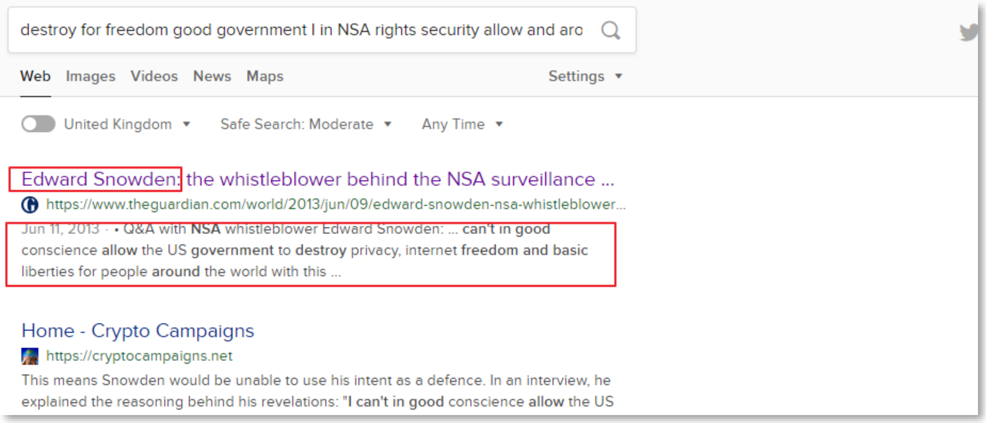
Figure 37

However, the file names seemed interesting and the second ZIP (ojd34.zip) was carved out similarly from Wireshark and after confirming it was a ZIP file using the previous methods, the same error was displayed again when attempting the unzipping process. This archive also had a set of 10 files which again looked like images based on file extensions. They were also named in a similar fashion, using what seemed like specific English words (Figure 38). After all 20 words were used as a search term, several hits about a quote by Edward Snowden was found, however, it seemed like several files or "words" were still missing (Figure 39).



Name	Size	Type	Modified
allow.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
and.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
around.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
basic.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
building.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
cant.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
conscience.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
terrorism.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
Watergate.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
web-based.jpg	5.4 kB	JPEG image	23 June 2014, 13:43

Figure 38



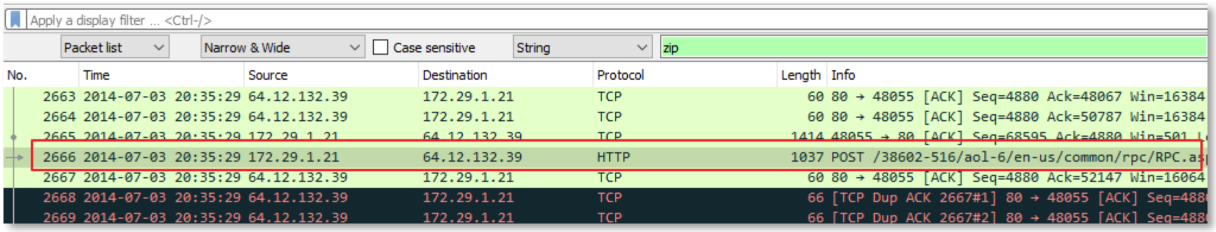
destroy for freedom good government I in NSA rights security allow and arc

Edward Snowden: the whistleblower behind the NSA surveillance ...
https://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower...
Jun 11, 2013 · • Q&A with NSA whistleblower Edward Snowden: ... can't in good conscience allow the US government to destroy privacy, internet freedom and basic liberties for people around the world with this ...

Home - Crypto Campaigns
https://cryptocampaigns.net
This means Snowden would be unable to use his intent as a defence. In an interview, he explained the reasoning behind his revelations: "I can't in good conscience allow the US

Figure 39

The PCAP file was then searched for any strings that included “zip” in them (Figure 40).

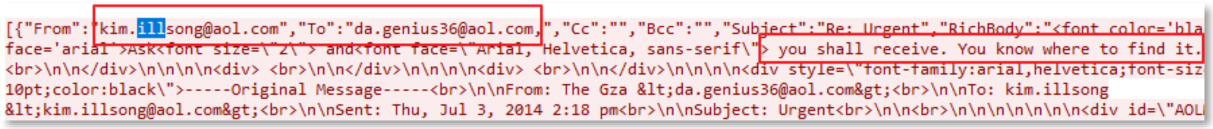


zip

```
No.   Time                Source         Destination    Protocol   Length Info
 2663 2014-07-03 20:35:29 64.12.132.39   172.29.1.21    TCP            60 80 → 48055 [ACK] Seq=4880 Ack=48067 Win=16384
 2664 2014-07-03 20:35:29 64.12.132.39   172.29.1.21    TCP            60 80 → 48055 [ACK] Seq=4880 Ack=50787 Win=16384
 2665 2014-07-03 20:35:29 172.29.1.21    64.12.132.39   TCP          1414 48055 → 80 [ACK] Seq=68595 Ack=4880 Win=501 L
 2666 2014-07-03 20:35:29 172.29.1.21    64.12.132.39   HTTP         1037 POST /38602-516/aol-6/en-us/common/rpc/RPC.as
 2667 2014-07-03 20:35:29 64.12.132.39   172.29.1.21    TCP            60 80 → 48055 [ACK] Seq=4880 Ack=52147 Win=16064
 2668 2014-07-03 20:35:29 64.12.132.39   172.29.1.21    TCP            66 [TCP Dup ACK 2667#1] 80 → 48055 [ACK] Seq=488
 2669 2014-07-03 20:35:29 64.12.132.39   172.29.1.21    TCP            66 [TCP Dup ACK 2667#2] 80 → 48055 [ACK] Seq=488
```

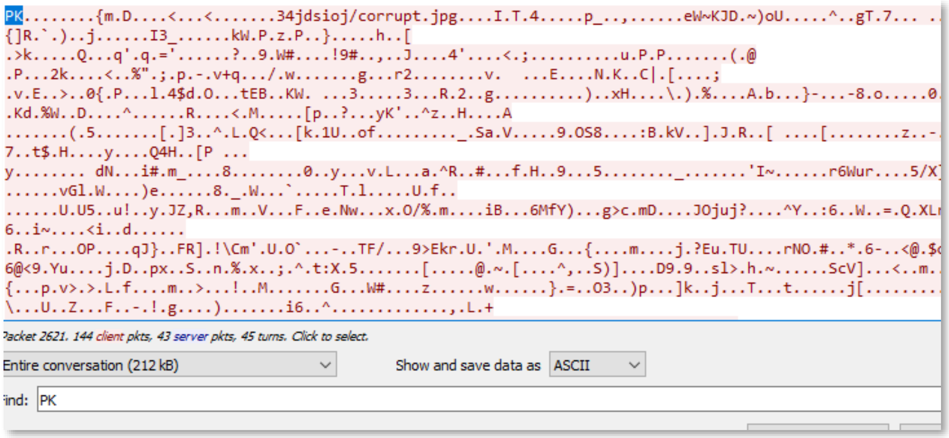
Figure 40

When the TCP stream was followed and the data was searched for known usernames, one result looked like an email from Ill-Song to a user named da.genius3@aol.com (Figure 41). Next, the bytes were searched for the characters “PK” which indicate a ZIP file and several hits were found (Figure 42). Based on the sender and recipient information of the found email, Ill-Song’s messages were the “red” messages and since the aim was to find out what they sent to the other user, only Ill-Song’s bytes were of interest at this point. The other bytes where filtered out and the remaining ones saved as raw hex like before.



```
[{"From":"kim.illsong@aol.com","To":"da.genius36@aol.com","Cc":"","Bcc":"","Subject":"Re: Urgent","RichBody":"<font color='bla
face='arial'>Ask<font size=\"2\"> and<font face=\"Arial, Helvetica, sans-serif\"> you shall receive. You know where to find it.
<br>\n\n</div>\n\n\n\n<div> <br>\n\n</div>\n\n\n\n<div> <br>\n\n</div>\n\n\n\n<div style=\"font-family:arial,helvetica;font-siz
10pt;color:black\">-----Original Message-----<br>\n\nFrom: The Gza &lt;da.genius36@aol.com&gt;<br>\n\nTo: kim.illsong
&lt;kim.illsong@aol.com&gt;<br>\n\nSent: Thu, Jul 3, 2014 2:18 pm<br>\n\nSubject: Urgent<br>\n\n<br>\n\n\n\n\n\n\n<div id=\"AOL
```

Figure 41



```
PK........{m.D....<...<.......34jdsioj/corrupt.jpg....I.T.4.....p_..,......eW~KJD.~)oU.....^..gT.7... ..
{]R.`.)..j......I3_......kW.P.z.P..}.....h..[
.>k.....Q...q'.q.='......?..9.W#....!9#..,..J....4'....<.;..........u.P.P.......(.@
.P...2k....<..%".;.p.-.v+q.../.w.......g...r2........v.  ...E....N.K..C|.[....;
.v.E..>..0{.P...l.4$d.O...tEB..KW. ...3.....3...R.2..g..........)..xH....\.).%....A.b...}-...-8.o.....0
.Kd.%W..D....^.......R....<.M.....[p..?...yK'..^z..H....A
.......(.5.......[.]3..^.L.Q<...[k.1U..of.........._.Sa.V.....9.OS8....:B.kV..].J.R..[ ....[........z..-
7..t$.H....y....Q4H..[P ...
y........ dN...i#.m_....8........0..y...v.L...a.^R..#...f.H..9...5........_.......'I~......r6Wur....5/X]
......vGl.W....)e......8._.W...`.....T.l.....U.f..
......U.U5..u!..y.JZ,R...m..V...F..e.Nw...x.O/%.m....iB...6MfY)...g>c.mD....JOjuj?....^Y..:6..W..=.Q.XLr
6..i~....<i..d......
.R..r...OP....qJ}..FR].!\Cm'.U.O`...-..TF/...9>Ekr.U.'.M....G...{....m....j.?Eu.TU....rNO.#..*.6-..<@.$c
6@<9.Yu....j.D..px..S..n.%.x..;.^.t:X.5.......[.....@.~.[....^,..S)]....D9.9..sl>.h.~......ScV]...<..m..
{...p.v>.>.L.f....m..>...!..M.......G...W#....z......w......}.=..O3..)p...]k..j...T...t......j[........
\...U..Z...F..-.!.g....).......i6..^.............,.L.+
```

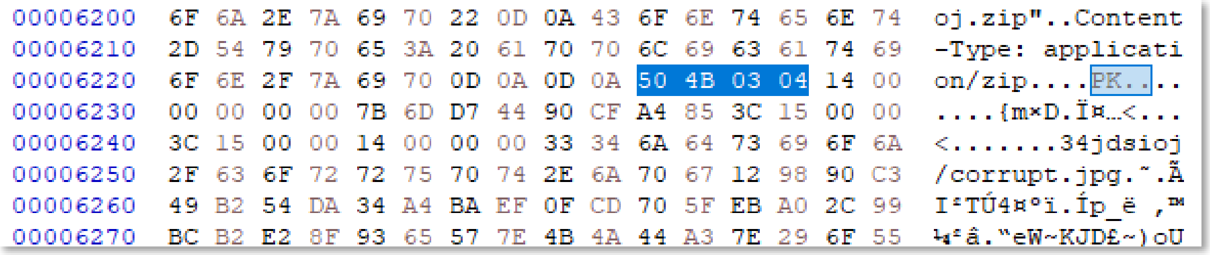
Packet 2621. 144 client pkts, 43 server pkts, 45 turns. Click to select.

Entire conversation (212 kB) Show and save data as ASCII

Find: PK

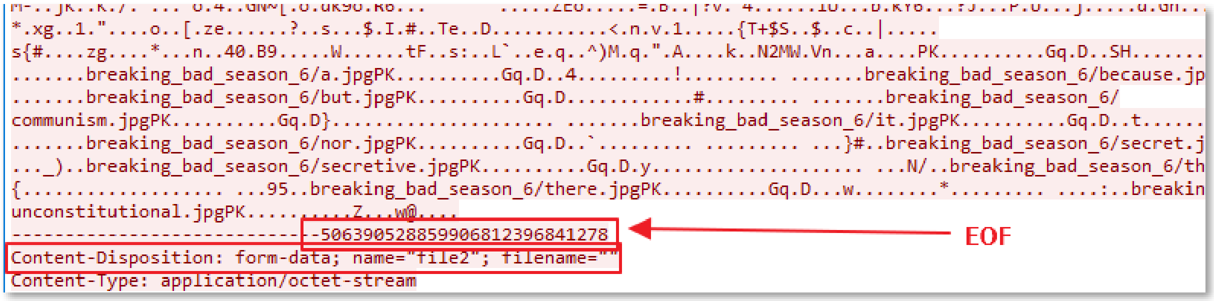
Figure 42

After this, the file was opened in a hex editor to carve out any ZIP files contained in it. The file was searched for all occurrences of the magic bytes *50 4B 03 04* which mark the beginning of a ZIP archive (Figure 43). The Wireshark TCP stream was examined to see how the file terminated and it seemed to have a long number as its last bytes (Figure 44). To find this in the hex editor, a search was done for the string "Content-Disposition" which was some kind of a header (Figure 45).



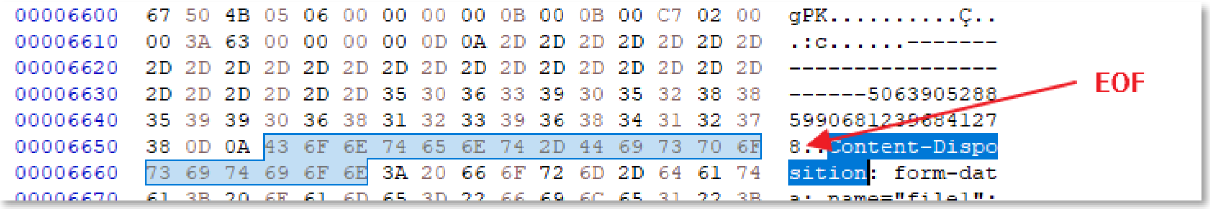
```
00006200  6F 6A 2E 7A 69 70 22 0D 0A 43 6F 6E 74 65 6E 74  oj.zip"..Content
00006210  2D 54 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69  -Type: applicati
00006220  6F 6E 2F 7A 69 70 0D 0A 0D 0A 50 4B 03 04 14 00  on/zip....PK....
00006230  00 00 00 00 7B 6D D7 44 90 CF A4 85 3C 15 00 00  ....{m×D.Ï¤…<...
00006240  3C 15 00 00 14 00 00 00 33 34 6A 64 73 69 6F 6A  <.......34jdsioj
00006250  2F 63 6F 72 72 75 70 74 2E 6A 70 67 12 98 90 C3  /corrupt.jpg.˜.Ã
00006260  49 B2 54 DA 34 A4 BA EF 0F CD 70 5F EB A0 2C 99  I²TÚ4¤ºï.Íp_ë ,™
00006270  BC B2 E2 8F 93 65 57 7E 4B 4A 44 A3 7E 29 6F 55  ¼²â.“eW~KJD£~)oU
```

Figure 43



```
*.xg..1."....o..[.ze......?..s...$.I.#..Te..D...........<.n.v.1.....{T+$S..$..c..|.....
s{#....zg....*...n..40.B9.....W......tF..s:..L`..e.q..^)M.q.".A....k..N2MW.Vn...a....PK..........Gq.D..SH.......
.......breaking_bad_season_6/a.jpgPK..........Gq.D..4.........!......... .......breaking_bad_season_6/because.jp
.......breaking_bad_season_6/but.jpgPK..........Gq.D............#......... .......breaking_bad_season_6/
communism.jpgPK..........Gq.D}..................... .......breaking_bad_season_6/it.jpgPK..........Gq.D..t.......
.......breaking_bad_season_6/nor.jpgPK..........Gq.D..`......... ......... ...}#..breaking_bad_season_6/secret.j
..._)..breaking_bad_season_6/secretive.jpgPK..........Gq.D.y.................... ...N/..breaking_bad_season_6/th
{.................. ...95..breaking_bad_season_6/there.jpgPK..........Gq.D...w........*......... ....:..breakin
unconstitutional.jpgPK..........Z...w@....
-----------------------------506390528859906812396841278
Content-Disposition: form-data; name="file2"; filename=""
Content-Type: application/octet-stream
```

Figure 44

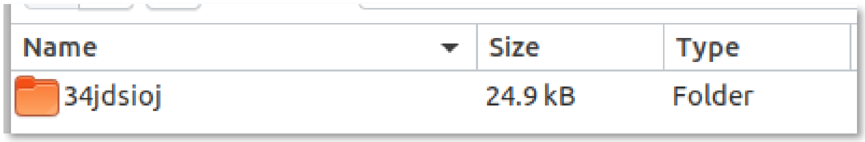


```
00006600  67 50 4B 05 06 00 00 00 00 0B 00 0B 00 C7 02 00  gPK..........Ç..
00006610  00 3A 63 00 00 00 00 0D 0A 2D 2D 2D 2D 2D 2D 2D  .:c......-------
00006620  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D  ----------------
00006630  2D 2D 2D 2D 2D 2D 35 30 36 33 39 30 35 32 38 38  ------5063905288
00006640  35 39 39 30 36 38 31 32 33 39 36 38 34 31 32 37  5990681239684127
00006650  38 0D 0A 43 6F 6E 74 65 6E 74 2D 44 69 73 70 6F  8..Content-Dispo
00006660  73 69 74 69 6F 6E 3A 20 66 6F 72 6D 2D 64 61 74  sition: form-dat
00006670  61 3B 20 6E 61 6D 65 3D 22 66 69 6C 65 31 22 3B  a; name="file1";
```

Figure 45

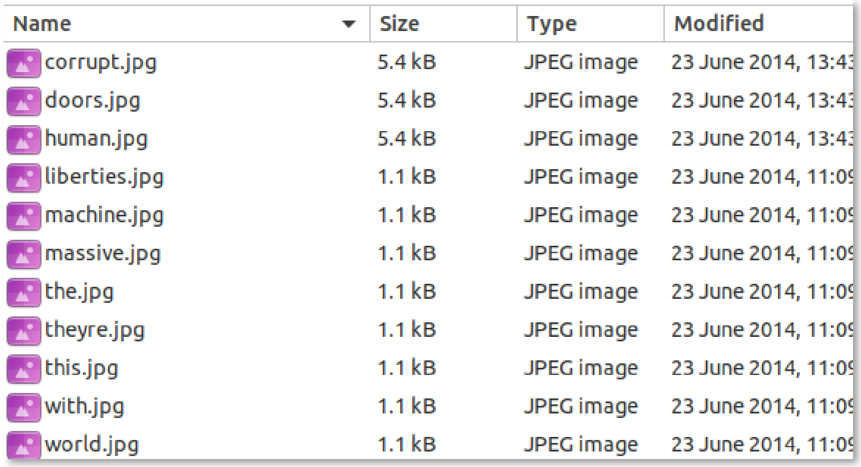
The bytes starting from the magic bytes up until the EOF were saved as its own file and confirmed to be a ZIP file using *file*. When viewed, it contained a folder called 34jdsioj (Figure 46) and it contained a set of 11 files named in a similar fashion as the previous ZIPs (Figure 47).

34jdsioj



Name	Size	Type
34jdsioj	24.9 kB	Folder

Figure 46

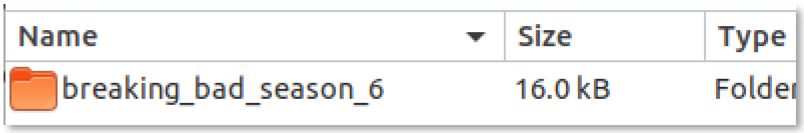


Name	Size	Type	Modified
corrupt.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
doors.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
human.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
liberties.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
machine.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
massive.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
the.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
theyre.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
this.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
with.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
world.jpg	1.1 kB	JPEG image	23 June 2014, 11:09

Figure 47

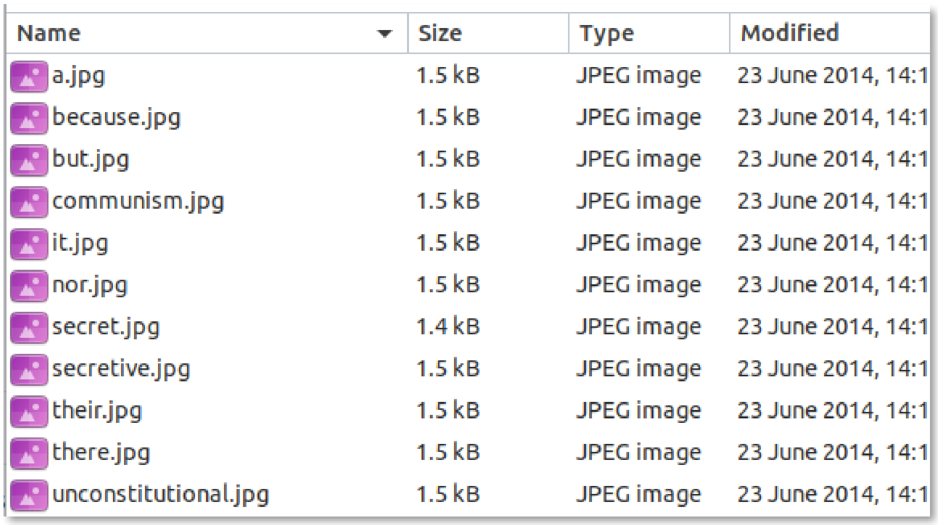
This process of searching for the magic bytes and the EOF was repeated and two more similar ZIP files were carved out (Figures 48-51).

breaking_bad_season_6



Name	Size	Type
breaking_bad_season_6	16.0 kB	Folder

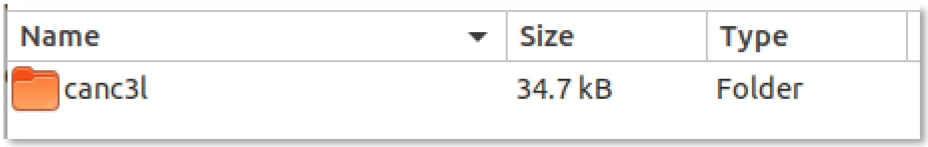
Figure 48



Name	Size	Type	Modified
a.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
because.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
but.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
communism.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
it.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
nor.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
secret.jpg	1.4 kB	JPEG image	23 June 2014, 14:1
secretive.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
their.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
there.jpg	1.5 kB	JPEG image	23 June 2014, 14:1
unconstitutional.jpg	1.5 kB	JPEG image	23 June 2014, 14:1

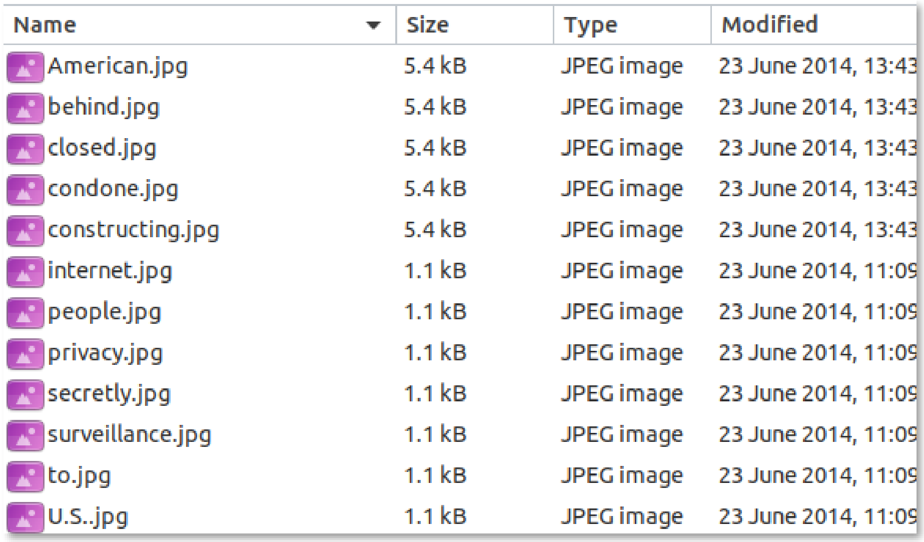
Figure 49

canc3l



Name	Size	Type
canc3l	34.7 kB	Folder

Figure 50

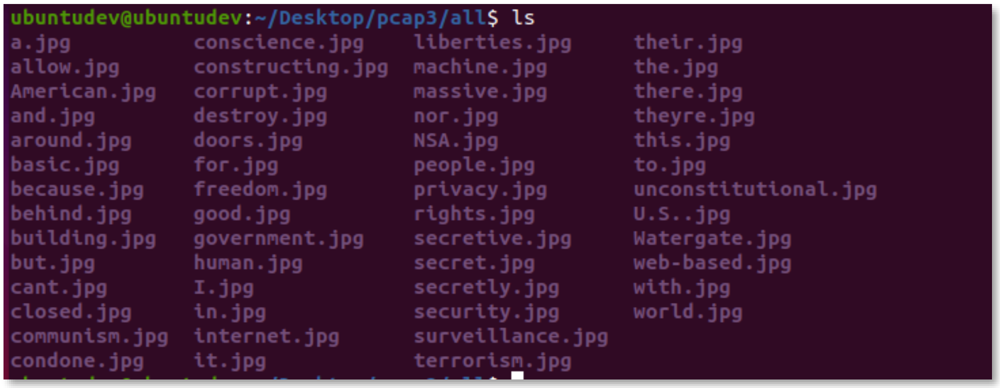


Name	Size	Type	Modified
American.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
behind.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
closed.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
condone.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
constructing.jpg	5.4 kB	JPEG image	23 June 2014, 13:43
internet.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
people.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
privacy.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
secretly.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
surveillance.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
to.jpg	1.1 kB	JPEG image	23 June 2014, 11:09
U.S..jpg	1.1 kB	JPEG image	23 June 2014, 11:09

Figure 51

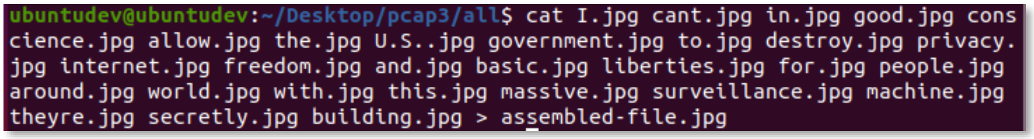
All the files in these three new ZIP files extracted perfectly and all files from the five ZIP archives were copied into a single folder (Figure 52). Using each file once, they were assembled into a single file based on the quote by Edward Snowden determining the order (Figure 53). The complete quote from an article by The Guardian:

> I'm willing to sacrifice all of that because I can't in good conscience allow the US government to destroy privacy, internet freedom and basic liberties for people around the world with this massive surveillance machine they're secretly building. (The Guardian, 2013)



```
ubuntudev@ubuntudev:~/Desktop/pcap3/all$ ls
a.jpg          conscience.jpg    liberties.jpg     their.jpg
allow.jpg      constructing.jpg  machine.jpg       the.jpg
American.jpg   corrupt.jpg       massive.jpg       there.jpg
and.jpg        destroy.jpg       nor.jpg           theyre.jpg
around.jpg     doors.jpg         NSA.jpg           this.jpg
basic.jpg      for.jpg           people.jpg        to.jpg
because.jpg    freedom.jpg       privacy.jpg       unconstitutional.jpg
behind.jpg     good.jpg          rights.jpg        U.S..jpg
building.jpg   government.jpg    secretive.jpg     Watergate.jpg
but.jpg        human.jpg         secret.jpg        web-based.jpg
cant.jpg       I.jpg             secretly.jpg      with.jpg
closed.jpg     in.jpg            security.jpg      world.jpg
communism.jpg  internet.jpg      surveillance.jpg
condone.jpg    it.jpg            terrorism.jpg
```

Figure 52



```
ubuntudev@ubuntudev:~/Desktop/pcap3/all$ cat I.jpg cant.jpg in.jpg good.jpg cons
cience.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 > assembled-file.jpg
```

Figure 53

The assembled file as a valid jpg of a fancy chess set which most likely was the intended bribe (Figure 54).

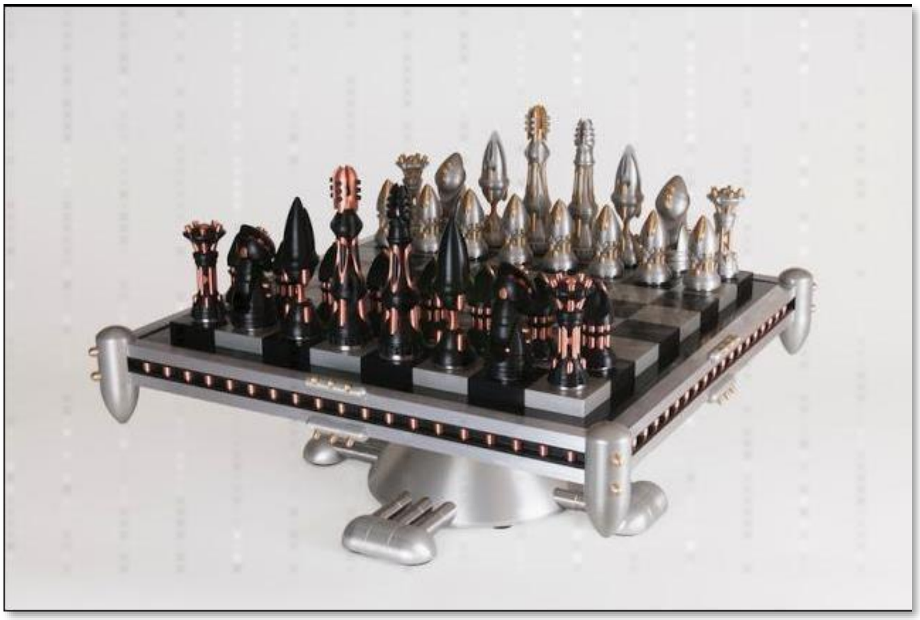


Figure 54

REFERENCES CAPTURE 3.PCAP

Gary Kessler Associates (2019) *GCK'S FILE SIGNATURES TABLE*. Available at: https://www.garykessler.net/library/file_sigs.html (Accessed: 8 December 2019).

The Guardian (2013) *Edward Snowden: the whistleblower behind the NSA surveillance revelations*. Available at: https://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance (Accessed: 8 December 2019).

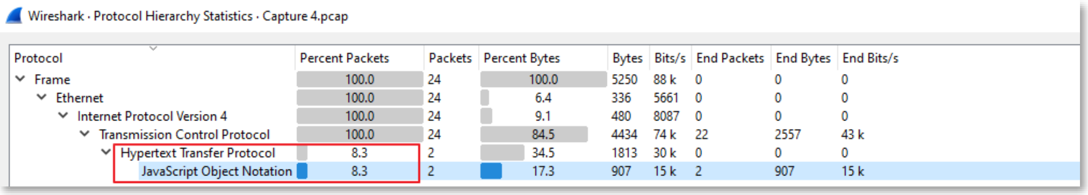
4 Investigation of Capture 4.pcap

4.1 Abstract

A PCAP file with what seemed like SMS messages between a user called Ann and Ill-Song was examined. The user Ann wanted to set up a meeting between the two and in the messages revealed the time (5PM) and month (September) for the meet. The PCAP also contained several GPS coordinates Ann posted from a map service API, and by overlaying several of these on a map, the number 17 was traced onto it most likely signifying the date for the meet.

4.2 Procedure

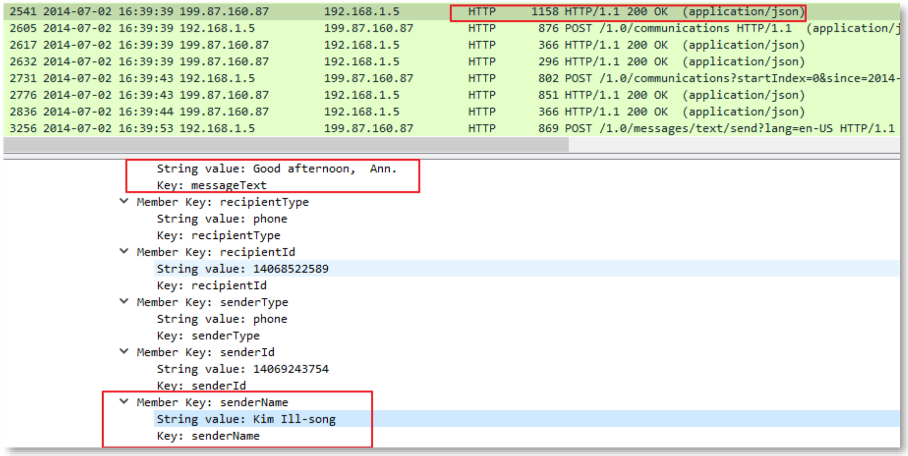
To start the investigation on the PCAP, the protocol hierarchy statistics were viewed (Figure 55). The breakdown showed several of the packets being HTTP/Javascript and once the JSON packets were applied as a filter, a manageable number of packets are displayed. The messages are transmitted in either HTTP 200 OK packets or in POST requests (Figure 57) and the metadata looked like they were SMS messages or something similar (Figure 58). An example content listing of such a packet has several useful fields set, such as senderName and messageTxt (Figure 56).



Wireshark · Protocol Hierarchy Statistics · Capture 4.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	24	100.0	5250	88 k	0	0	0
Ethernet	100.0	24	6.4	336	5661	0	0	0
Internet Protocol Version 4	100.0	24	9.1	480	8087	0	0	0
Transmission Control Protocol	100.0	24	84.5	4434	74 k	22	2557	43 k
Hypertext Transfer Protocol	8.3	2	34.5	1813	30 k	0	0	0
JavaScript Object Notation	8.3	2	17.3	907	15 k	2	907	15 k

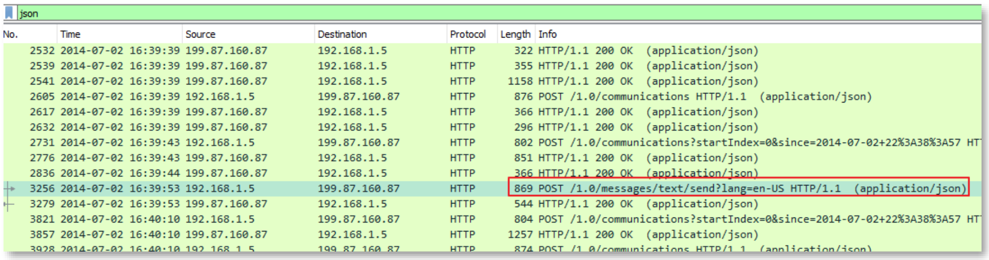
Figure 55



```
2541 2014-07-02 16:39:39 199.87.160.87    192.168.1.5      HTTP    1158 HTTP/1.1 200 OK  (application/json)
2605 2014-07-02 16:39:39 192.168.1.5      199.87.160.87    HTTP     876 POST /1.0/communications HTTP/1.1  (application/j
2617 2014-07-02 16:39:39 199.87.160.87    192.168.1.5      HTTP     366 HTTP/1.1 200 OK  (application/json)
2632 2014-07-02 16:39:39 199.87.160.87    192.168.1.5      HTTP     296 HTTP/1.1 200 OK  (application/json)
2731 2014-07-02 16:39:43 192.168.1.5      199.87.160.87    HTTP     802 POST /1.0/communications?startIndex=0&since=2014-
2776 2014-07-02 16:39:43 199.87.160.87    192.168.1.5      HTTP     851 HTTP/1.1 200 OK  (application/json)
2836 2014-07-02 16:39:44 199.87.160.87    192.168.1.5      HTTP     366 HTTP/1.1 200 OK  (application/json)
3256 2014-07-02 16:39:53 192.168.1.5      199.87.160.87    HTTP     869 POST /1.0/messages/text/send?lang=en-US HTTP/1.1
```

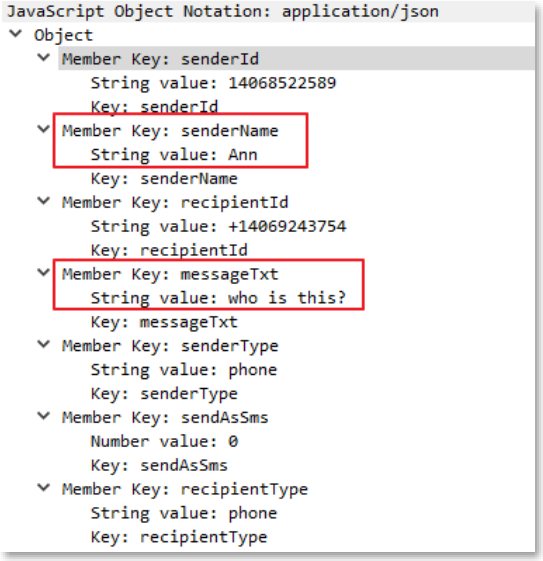
```
      String value: Good afternoon,  Ann.
      Key: messageText
  Member Key: recipientType
      String value: phone
      Key: recipientType
  Member Key: recipientId
      String value: 14068522589
      Key: recipientId
  Member Key: senderType
      String value: phone
      Key: senderType
  Member Key: senderId
      String value: 14069243754
      Key: senderId
  Member Key: senderName
      String value: Kim Ill-song
      Key: senderName
```

Figure 56



```
json
No.     Time                Source          Destination     Protocol  Length  Info
   2532 2014-07-02 16:39:39 199.87.160.87   192.168.1.5     HTTP       322 HTTP/1.1 200 OK  (application/json)
   2539 2014-07-02 16:39:39 199.87.160.87   192.168.1.5     HTTP       355 HTTP/1.1 200 OK  (application/json)
   2541 2014-07-02 16:39:39 199.87.160.87   192.168.1.5     HTTP      1158 HTTP/1.1 200 OK  (application/json)
   2605 2014-07-02 16:39:39 192.168.1.5     199.87.160.87   HTTP       876 POST /1.0/communications HTTP/1.1  (application/json)
   2617 2014-07-02 16:39:39 199.87.160.87   192.168.1.5     HTTP       366 HTTP/1.1 200 OK  (application/json)
   2632 2014-07-02 16:39:39 199.87.160.87   192.168.1.5     HTTP       296 HTTP/1.1 200 OK  (application/json)
   2731 2014-07-02 16:39:43 192.168.1.5     199.87.160.87   HTTP       802 POST /1.0/communications?startIndex=0&since=2014-07-02+22%3A38%3A57 HT
   2776 2014-07-02 16:39:43 199.87.160.87   192.168.1.5     HTTP       851 HTTP/1.1 200 OK  (application/json)
   2836 2014-07-02 16:39:44 199.87.160.87   192.168.1.5     HTTP       366 HTTP/1.1 200 OK  (application/json)
   3256 2014-07-02 16:39:53 192.168.1.5     199.87.160.87   HTTP       869 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
   3279 2014-07-02 16:39:53 199.87.160.87   192.168.1.5     HTTP       544 HTTP/1.1 200 OK  (application/json)
   3821 2014-07-02 16:40:10 192.168.1.5     199.87.160.87   HTTP       804 POST /1.0/communications?startIndex=0&since=2014-07-02+22%3A38%3A57 HT
   3857 2014-07-02 16:40:10 199.87.160.87   192.168.1.5     HTTP      1257 HTTP/1.1 200 OK  (application/json)
   3928 2014-07-02 16:40:10 192.168.1.5     199.87.160.87   HTTP       874 POST /1.0/communications HTTP/1.1  (application/json)
```

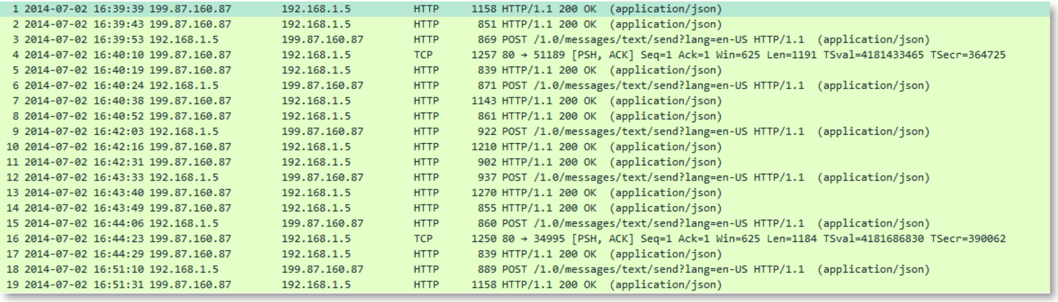
Figure 57



```
JavaScript Object Notation: application/json
 ˅ Object
   ˅ Member Key: senderId
       String value: 14068522589
       Key: senderId
   ˅ Member Key: senderName
       String value: Ann
       Key: senderName
   ˅ Member Key: recipientId
       String value: +14069243754
       Key: recipientId
   ˅ Member Key: messageTxt
       String value: who is this?
       Key: messageTxt
   ˅ Member Key: senderType
       String value: phone
       Key: senderType
   ˅ Member Key: sendAsSms
       Number value: 0
       Key: sendAsSms
   ˅ Member Key: recipientType
       String value: phone
       Key: recipientType
```

Figure 58

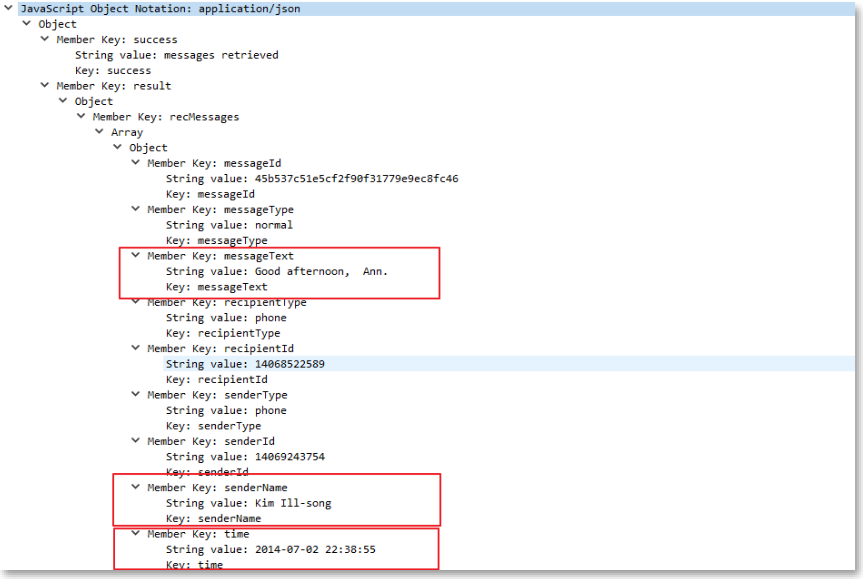
To further filter the packets, *tshark* was used to find packets that had a "senderName" field and matched the display filter 'json'. These packets were then saved into a new PCAP and it had a total of 19 packets which were easy to analyse further (Figure 59).



```
 1 2014-07-02 16:39:39 199.87.160.87    192.168.1.5      HTTP     1158 HTTP/1.1 200 OK  (application/json)
 2 2014-07-02 16:39:43 199.87.160.87    192.168.1.5      HTTP      851 HTTP/1.1 200 OK  (application/json)
 3 2014-07-02 16:39:53 192.168.1.5      199.87.160.87    HTTP      869 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
 4 2014-07-02 16:40:10 199.87.160.87    192.168.1.5      TCP      1257 80 → 51189 [PSH, ACK] Seq=1 Ack=1 Win=625 Len=1191 TSval=4181433465 TSecr=364725
 5 2014-07-02 16:40:19 199.87.160.87    192.168.1.5      HTTP      839 HTTP/1.1 200 OK  (application/json)
 6 2014-07-02 16:40:24 192.168.1.5      199.87.160.87    HTTP      871 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
 7 2014-07-02 16:40:38 199.87.160.87    192.168.1.5      HTTP     1143 HTTP/1.1 200 OK  (application/json)
 8 2014-07-02 16:40:52 199.87.160.87    192.168.1.5      HTTP      861 HTTP/1.1 200 OK  (application/json)
 9 2014-07-02 16:42:03 192.168.1.5      199.87.160.87    HTTP      922 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
10 2014-07-02 16:42:16 199.87.160.87    192.168.1.5      HTTP     1210 HTTP/1.1 200 OK  (application/json)
11 2014-07-02 16:42:31 199.87.160.87    192.168.1.5      HTTP      902 HTTP/1.1 200 OK  (application/json)
12 2014-07-02 16:43:33 192.168.1.5      199.87.160.87    HTTP      937 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
13 2014-07-02 16:43:40 199.87.160.87    192.168.1.5      HTTP     1270 HTTP/1.1 200 OK  (application/json)
14 2014-07-02 16:43:49 199.87.160.87    192.168.1.5      HTTP      855 HTTP/1.1 200 OK  (application/json)
15 2014-07-02 16:44:06 192.168.1.5      199.87.160.87    HTTP      860 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
16 2014-07-02 16:44:23 199.87.160.87    192.168.1.5      TCP      1250 80 → 34995 [PSH, ACK] Seq=1 Ack=1 Win=625 Len=1184 TSval=4181686830 TSecr=390062
17 2014-07-02 16:44:29 199.87.160.87    192.168.1.5      HTTP      839 HTTP/1.1 200 OK  (application/json)
18 2014-07-02 16:51:10 192.168.1.5      199.87.160.87    HTTP      889 POST /1.0/messages/text/send?lang=en-US HTTP/1.1  (application/json)
19 2014-07-02 16:51:31 199.87.160.87    192.168.1.5      HTTP     1158 HTTP/1.1 200 OK  (application/json)
```

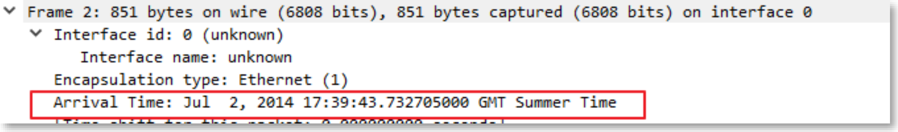
Figure 59

After this, each packet was analysed and the following parts from each packet were transcribed. senderName (sender), messageText (message) and time (time) (Figure 60). Because the messages that were received as a POST request did not have a json time value, the frame arrival time was used (Figure 61).



```
JavaScript Object Notation: application/json
  Object
    Member Key: success
      String value: messages retrieved
      Key: success
    Member Key: result
      Object
        Member Key: recMessages
          Array
            Object
              Member Key: messageId
                String value: 45b537c51e5cf2f90f31779e9ec8fc46
                Key: messageId
              Member Key: messageType
                String value: normal
                Key: messageType
              Member Key: messageText
                String value: Good afternoon,  Ann.
                Key: messageText
              Member Key: recipientType
                String value: phone
                Key: recipientType
              Member Key: recipientId
                String value: 14068522589
                Key: recipientId
              Member Key: senderType
                String value: phone
                Key: senderType
              Member Key: senderId
                String value: 14069243754
                Key: senderId
              Member Key: senderName
                String value: Kim Ill-song
                Key: senderName
              Member Key: time
                String value: 2014-07-02 22:38:55
                Key: time
```

Figure 60



```
Frame 2: 851 bytes on wire (6808 bits), 851 bytes captured (6808 bits) on interface 0
  Interface id: 0 (unknown)
      Interface name: unknown
  Encapsulation type: Ethernet (1)
  Arrival Time: Jul  2, 2014 17:39:43.732705000 GMT Summer Time
  [Time shift for this packet: 0.000000000 seconds]
```

Figure 61

Transcription

Message #1

Time: 2014-07-02 17:39:43

Sender: **Kim Ill-song**

Message: ***Good afternoon, Ann.***

Message #2

Time: 2014-07-02 17:39:53

Sender: **Ann**

Message: ***who is this?***

Message #3

Time: 2014-07-02 17:40:19

Sender: **Kim Ill-Song**

Message: ***Castling.***

Message #4

Time: 2014-07-02 17:40:24

Sender: **Ann**

Message: ***where are you?***

Message #5

Time: 2014-07-02 17:40:52

Sender: **Kim Ill-Song**

Message: ***I know I can't tell you that.***

Message #6

Time: 2014-07-02 17:42:03

Sender: **Ann**

Message: ***Do you know that there are people investigating Kim Ill-Song?***

Message #7

Time: 2014-07-02 17:42:31

Sender: **Kim Ill-Song**

Message: ***Of course. However, they will never know it is me behind the bribes.***

Message #8

Time: 2014-07-02 17:43:33

Sender: **Ann**

Message: ***still we should be careful. Pay attention. I want to meet in September at 5PM.***

Message #9

Time: 2014-07-02 17:43:49

Sender: **Kim Ill-Song**

Message: ***At our old meetup spot?***

Message #10

Time: 2014-07-02 17:44:06

Sender: **Ann**

Message: ***yes***

Message #11

Time: 2014-07-02 17:44:29

Sender: **Kim Ill-Song**

Message: ***What day?***

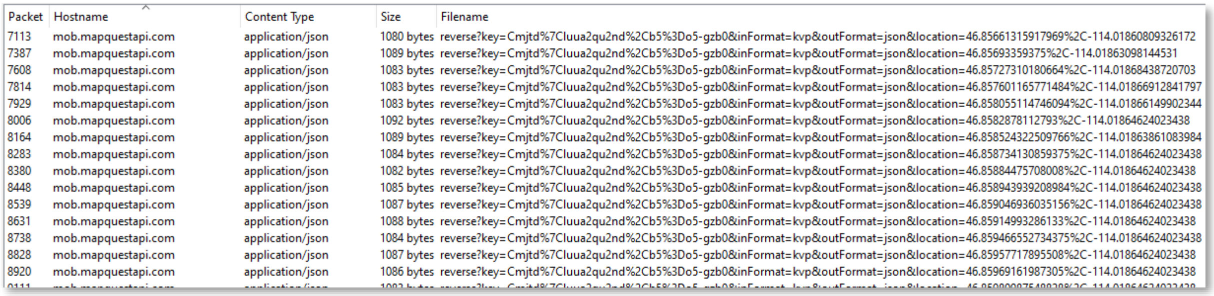
Message #12

Time: 2014-07-02 17:51:10

Sender: **Ann**

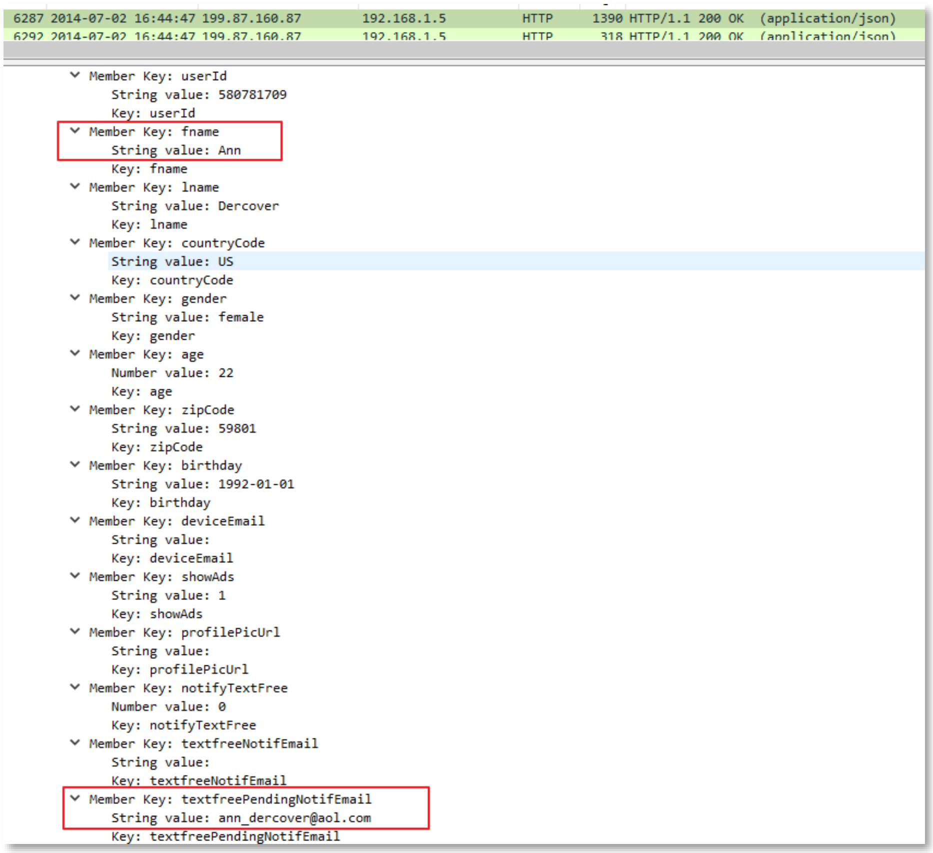
Message: ***I told you to pay attention.***

In message #8 Ann revealed the day and month of the meet and based on messages #11 and #12 the day was revealed around that time as well. However, as it's not present in the messages, the HTTP object list (File -> Export Objects -> HTTP) was examined. Once filtered based on name and skimmed through, there were several packets with the hostname mob.mapquestapi.com which hints towards a map/location service based on the name and what seemed like GPS coordinates in the URL. By using the text filter with "mapquestapi" all of these packets and their frame numbers were shown (Figure 62). Using the json display filter again, packet 6287 showed Ann interacting with mapquestapi which was further proof of them using an online map (Figure 63).



Packet	Hostname	Content Type	Size	Filename
7113	mob.mapquestapi.com	application/json	1080 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85661315917969%2C-114.01860809326172
7387	mob.mapquestapi.com	application/json	1089 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85693359375%2C-114.01863098144531
7608	mob.mapquestapi.com	application/json	1083 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85727310180664%2C-114.01868438720703
7814	mob.mapquestapi.com	application/json	1083 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.857601165771484%2C-114.01866912841797
7929	mob.mapquestapi.com	application/json	1083 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.858055114746094%2C-114.01866149902344
8006	mob.mapquestapi.com	application/json	1092 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.8582878112793%2C-114.01864624023438
8164	mob.mapquestapi.com	application/json	1089 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.858524322509766%2C-114.01863861083984
8283	mob.mapquestapi.com	application/json	1084 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.858734130859375%2C-114.01864624023438
8380	mob.mapquestapi.com	application/json	1082 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85884475708008%2C-114.01864624023438
8448	mob.mapquestapi.com	application/json	1085 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.858943939208984%2C-114.01864624023438
8539	mob.mapquestapi.com	application/json	1087 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.859046936035156%2C-114.01864624023438
8631	mob.mapquestapi.com	application/json	1088 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85914993286133%2C-114.01864624023438
8738	mob.mapquestapi.com	application/json	1084 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.859466552734375%2C-114.01864624023438
8828	mob.mapquestapi.com	application/json	1087 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85957717895508%2C-114.01864624023438
8920	mob.mapquestapi.com	application/json	1086 bytes	reverse?key=Cmjtd%7Cluua2qu2nd%2Cb5%3Do5-gzb0&inFormat=kvp&outFormat=json&location=46.85969161987305%2C-114.01864624023438

Figure 62



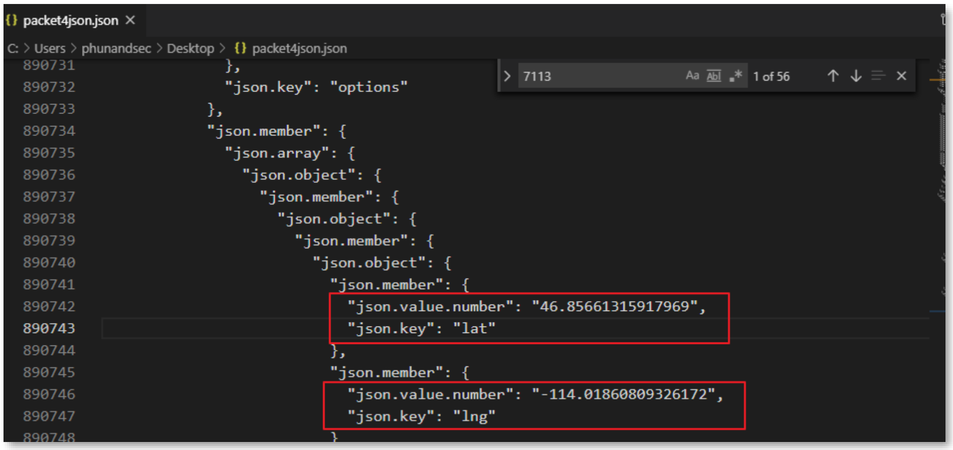
```
6287 2014-07-02 16:44:47 199.87.160.87       192.168.1.5       HTTP     1390 HTTP/1.1 200 OK  (application/json)
6292 2014-07-02 16:44:47 199.87.160.87       192.168.1.5       HTTP      318 HTTP/1.1 200 OK  (application/json)

      Member Key: userId
         String value: 580781709
         Key: userId
      Member Key: fname
         String value: Ann
         Key: fname
      Member Key: lname
         String value: Dercover
         Key: lname
      Member Key: countryCode
         String value: US
         Key: countryCode
      Member Key: gender
         String value: female
         Key: gender
      Member Key: age
         Number value: 22
         Key: age
      Member Key: zipCode
         String value: 59801
         Key: zipCode
      Member Key: birthday
         String value: 1992-01-01
         Key: birthday
      Member Key: deviceEmail
         String value:
         Key: deviceEmail
      Member Key: showAds
         String value: 1
         Key: showAds
      Member Key: profilePicUrl
         String value:
         Key: profilePicUrl
      Member Key: notifyTextFree
         Number value: 0
         Key: notifyTextFree
      Member Key: textfreeNotifEmail
         String value:
         Key: textfreeNotifEmail
      Member Key: textfreePendingNotifEmail
         String value: ann_dercover@aol.com
         Key: textfreePendingNotifEmail
```

Figure 63

Next the PCAP was exported as a JSON file using *tshark*.

```
tshark.exe -nr 'C:\Path\To\PCAP -T json > C:\Path\To\Output
```

The JSON file was analysed and searched for the first frame which included GPS data found in the HTTP object list search (Figure 62). This frame (7113) was examined more closely and the JSON keys for lat (latitude) and lng (longitude) where found (Figure 64).

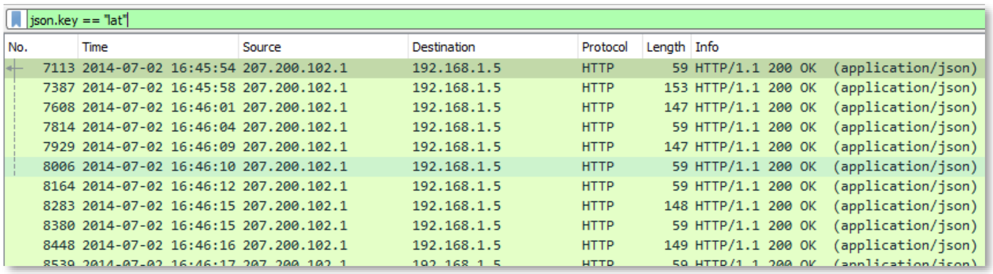


```
                },
                "json.key": "options"
              },
              "json.member": {
                "json.array": {
                  "json.object": {
                    "json.member": {
                      "json.object": {
                        "json.member": {
                          "json.object": {
                            "json.member": {
                              "json.value.number": "46.85661315917969",
                              "json.key": "lat"
                            },
                            "json.member": {
                              "json.value.number": "-114.01860809326172",
                              "json.key": "lng"
                            }
```

Figure 64

The json.key was found to be a display filter which showed all the packets that included GPS data (Figure 65). To extract the coordinates, tshark was used to parse them out (Figure 66):

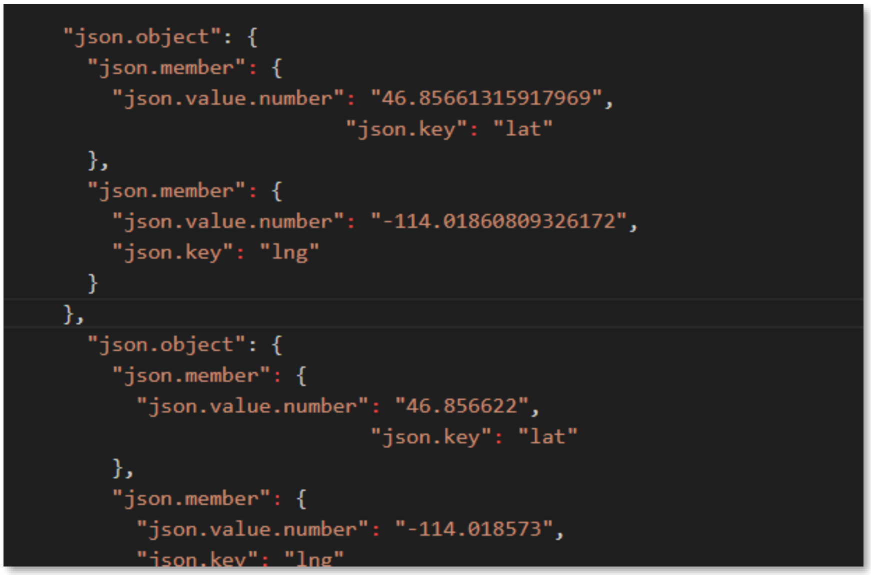
```
sls '"json.key": "lat"' .\packet4json.json -Context 3,6 > ./gpsData.json
```



```
json.key == "lat"
```

```
No.  Time                Source         Destination  Protocol Length Info
7113 2014-07-02 16:45:54 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
7387 2014-07-02 16:45:58 207.200.102.1  192.168.1.5  HTTP     153 HTTP/1.1 200 OK  (application/json)
7608 2014-07-02 16:46:01 207.200.102.1  192.168.1.5  HTTP     147 HTTP/1.1 200 OK  (application/json)
7814 2014-07-02 16:46:04 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
7929 2014-07-02 16:46:09 207.200.102.1  192.168.1.5  HTTP     147 HTTP/1.1 200 OK  (application/json)
8006 2014-07-02 16:46:10 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
8164 2014-07-02 16:46:12 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
8283 2014-07-02 16:46:15 207.200.102.1  192.168.1.5  HTTP     148 HTTP/1.1 200 OK  (application/json)
8380 2014-07-02 16:46:15 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
8448 2014-07-02 16:46:16 207.200.102.1  192.168.1.5  HTTP     149 HTTP/1.1 200 OK  (application/json)
8539 2014-07-02 16:46:17 207.200.102.1  192.168.1.5  HTTP      59 HTTP/1.1 200 OK  (application/json)
```

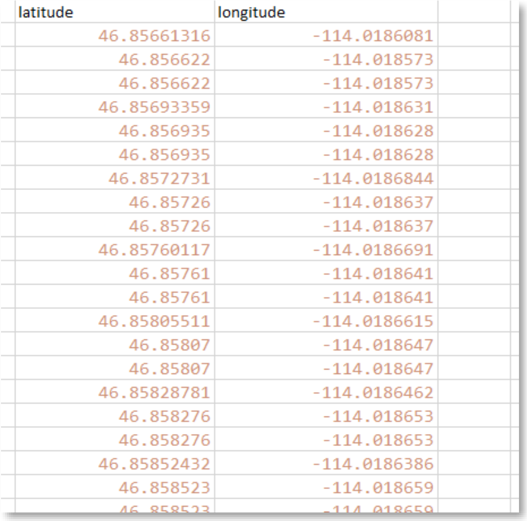
Figure 65



```
"json.object": {
  "json.member": {
    "json.value.number": "46.85661315917969",
                     "json.key": "lat"
  },
  "json.member": {
    "json.value.number": "-114.01860809326172",
    "json.key": "lng"
  }
},
  "json.object": {
    "json.member": {
      "json.value.number": "46.856622",
                       "json.key": "lat"
    },
    "json.member": {
      "json.value.number": "-114.018573",
      "json.key": "lng"
```

Figure 66

These latitude and longitude points were then written in a csv file (Figure 67), which was imported into Google Maps to plot the number 17 on a map (Figure 68).



latitude	longitude
46.85661316	-114.0186081
46.856622	-114.018573
46.856622	-114.018573
46.85693359	-114.018631
46.856935	-114.018628
46.856935	-114.018628
46.8572731	-114.0186844
46.85726	-114.018637
46.85726	-114.018637
46.85760117	-114.0186691
46.85761	-114.018641
46.85761	-114.018641
46.85805511	-114.0186615
46.85807	-114.018647
46.85807	-114.018647
46.85828781	-114.0186462
46.858276	-114.018653
46.858276	-114.018653
46.85852432	-114.0186386
46.858523	-114.018659
46.858523	-114.018659

Figure 67

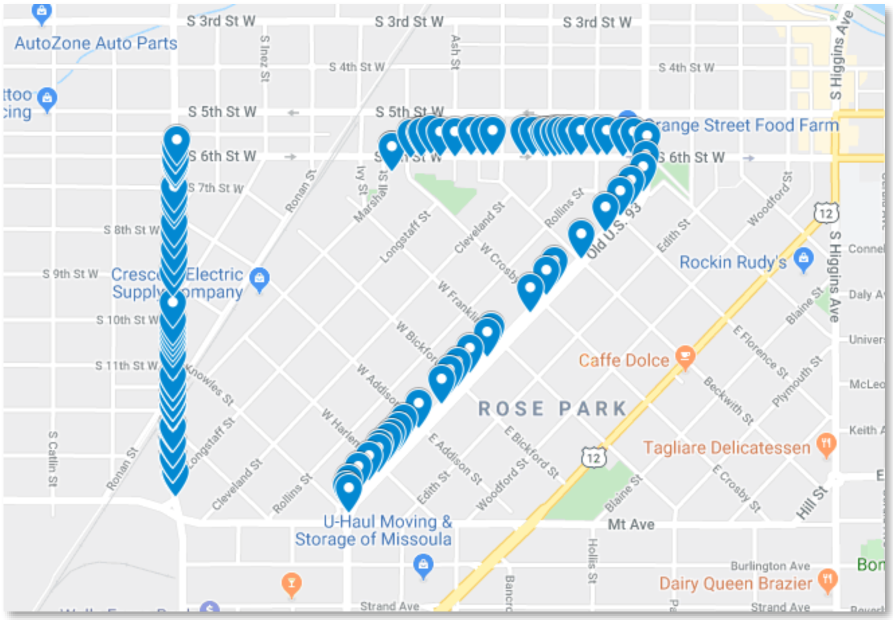


Figure 68

Based on the map and the conversation, the meet was on the **17th of September 2014** (year based on conversation year) at **5PM**.

REFERENCES CAPTURE 4.PCAP

Communary (2014) *Grep, the PowerShell way*. Available at: https://communary.net/2014/11/10/grep-the-powershell-way/ (Accessed: 6 December 2019).

Wireshark (2019) *Command line tshark JSON and Packet details all expanded*. Available at: https://ask.wireshark.org/question/12850/command-line-tshark-json-and-packet-details-all-expanded/ (Accessed: 6 December 2019).