Packet Analysis

By Brian Brown

NetSec

Syllabus: https://ubnetdef.org/courses/netsec/

- Ran by: Chris Crawford (DoD)

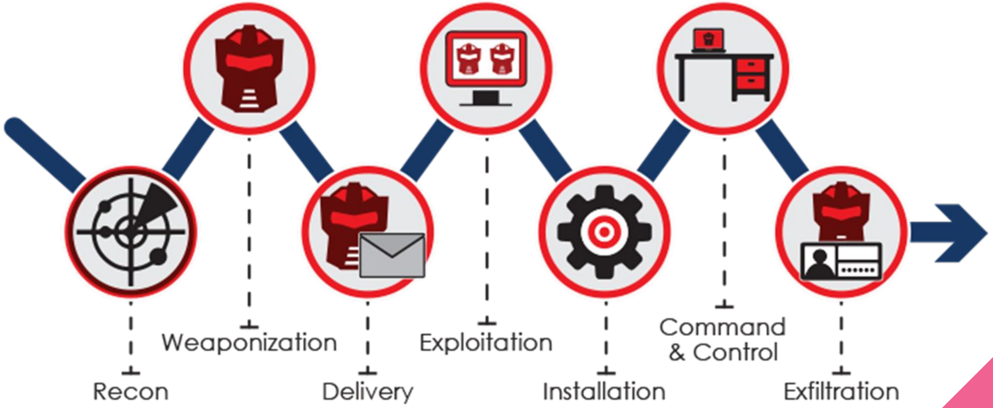
- @zachtenenbaum and @srini are TAs

What is Packet Analysis?

- Packet Analysis is the capture and interpretation of the traffic that occurs in your network.
- This includes capturing and recording traffic as it happens live.
- This also includes analyzing captured data and interpreting what it all means.
- For example: If a company has a compromised machine, they would perform a packet analysis to develop a storyline of who was infected, how they were infected, what were they infected with, and who attacked them.

Packet Analysis and Kill Chain

- Packet Analysis can be crucial in identifying multiple stages of the Kill Chain.
- By identifying these stages, it becomes easier to defend against an attacker at different stages of the Kill Chain.

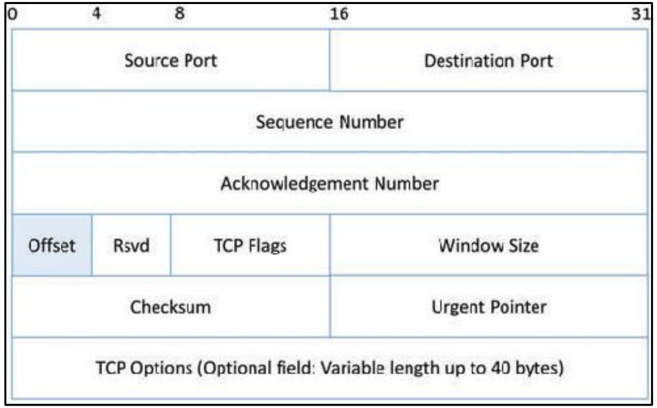


What is a packet?

- Wikipedia Definition: “A packet consists of control information and user data, which is also known as the payload. Control information provides data for delivering the payload, for example: source and destination network addresses, error detection codes, and sequencing information.”
- Think of it like an email or text message.
- Contains: Sender, Receiver, Contents.

Headers

- General: Contains information needed in order for a connection to be made such as the host and destination.
- TCP Header: Contains information to verify the packet for the three way TCP handshake.
  - Checksum: Used for error-checking header and payload.
  - Urgent Pointer: offset from the sequence number indicating the last urgent data byte.
  - TCP Flags: NS, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN.



0 4 8 16 31

Source Port | Destination Port

Sequence Number

Acknowledgement Number

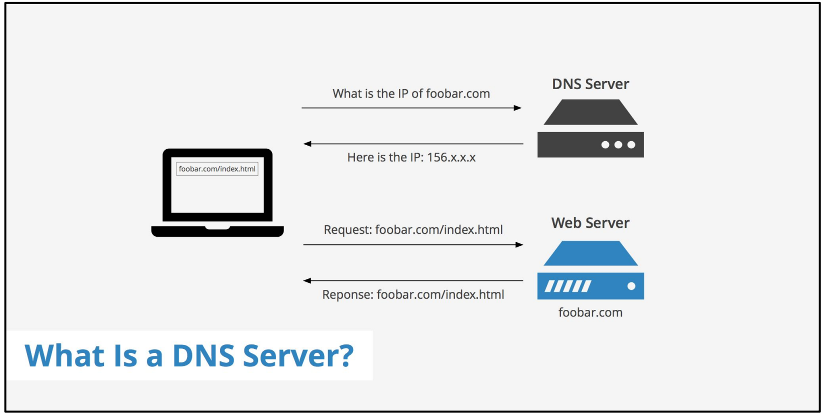
Offset | Rsvd | TCP Flags | Window Size

Checksum | Urgent Pointer

TCP Options (Optional field: Variable length up to 40 bytes)

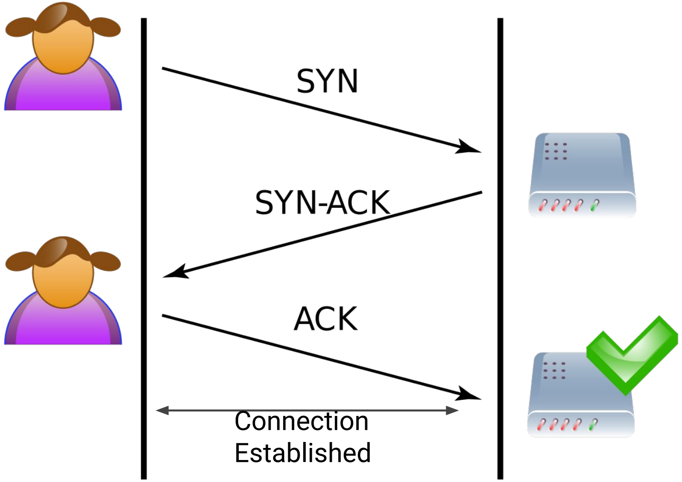
DNS

- Uses UDP instead of TCP to transport.
- Translates more readily memorized domain names to the numerical IP.
- For example: When you go to the website google.com, it navigates to the IP address 172.217.164.174.


foobar.com/index.html
What is the IP of foobar.com
DNS Server
Here is the IP: 156.x.x.x
Request: foobar.com/index.html
Web Server
Reponse: foobar.com/index.html
foobar.com
What Is a DNS Server?

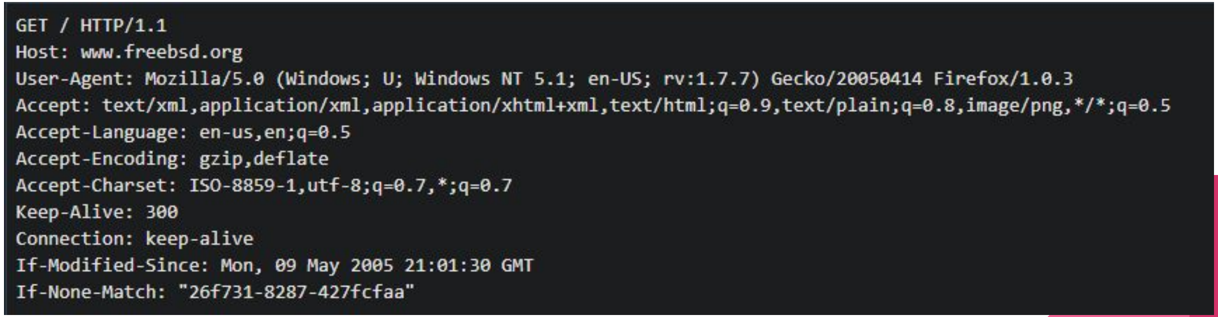
TCP

- Threeway Handshake: Used by TCP in order to establish a connection between the Host and Destination. Consists of 3 TCP Flags:
  - SYN
  - ACK
  - SYN & ACK
- Transport level of OSI



HTTP

- Multiple requests can be sent in one packet without waiting for the server’s response because HTTP used after TCP connection established.
- Requests are sent in plain text.
- Application level of the OSI model.



```
GET / HTTP/1.1
Host: www.freebsd.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.7) Gecko/20050414 Firefox/1.0.3
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
If-Modified-Since: Mon, 09 May 2005 21:01:30 GMT
If-None-Match: "26f731-8287-427fcfaa"
```

BREAK

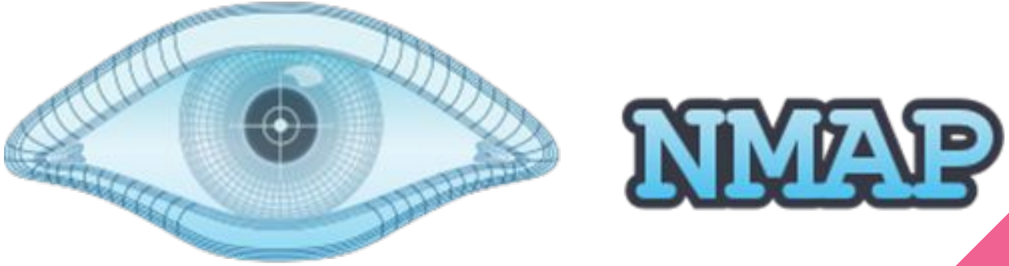
- Take a 15 minute break before we get to the fun stuff!

Packet Sniffing

- The process of gathering, collecting and logging packets in a network.
- WARNING: Be aware of environment you are sniffing in. You can get in trouble if you are sniffing in the wrong places (curiosity got the cat arrested).
- Sniffing can be used by both attackers and defenders.

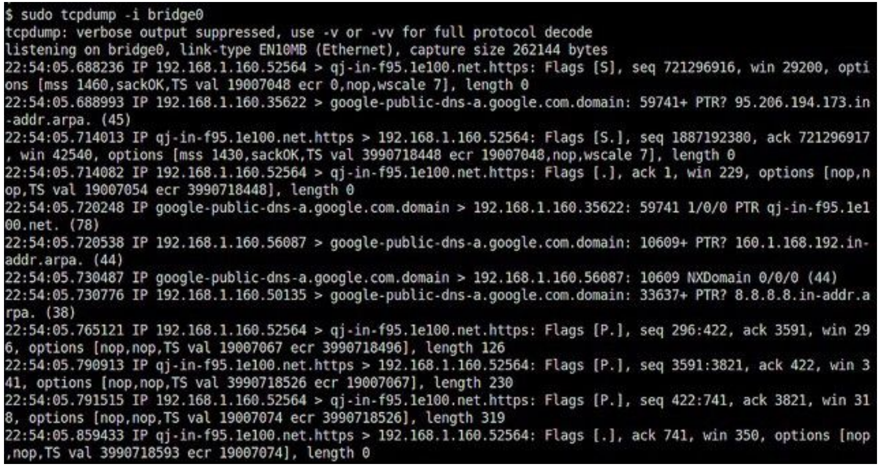
Network Mapper (Nmap)

- Nmap is a network analyzer that is primarily used for port scanning and Host Discovery.
- Nmap can be leveraged to capture network traffic as well to be analyze.
- https://youtu.be/HRmCe9ZLNUY?t=7
- Interested blog post: https://blog.webernetz.net/nmap-packet-capture/



Tcpdump

- A simple packet analyzer that utilizes the command line.
- Can read live traffic from the network or from a Packet Capture file.
- Prints out to the terminal or to a file.



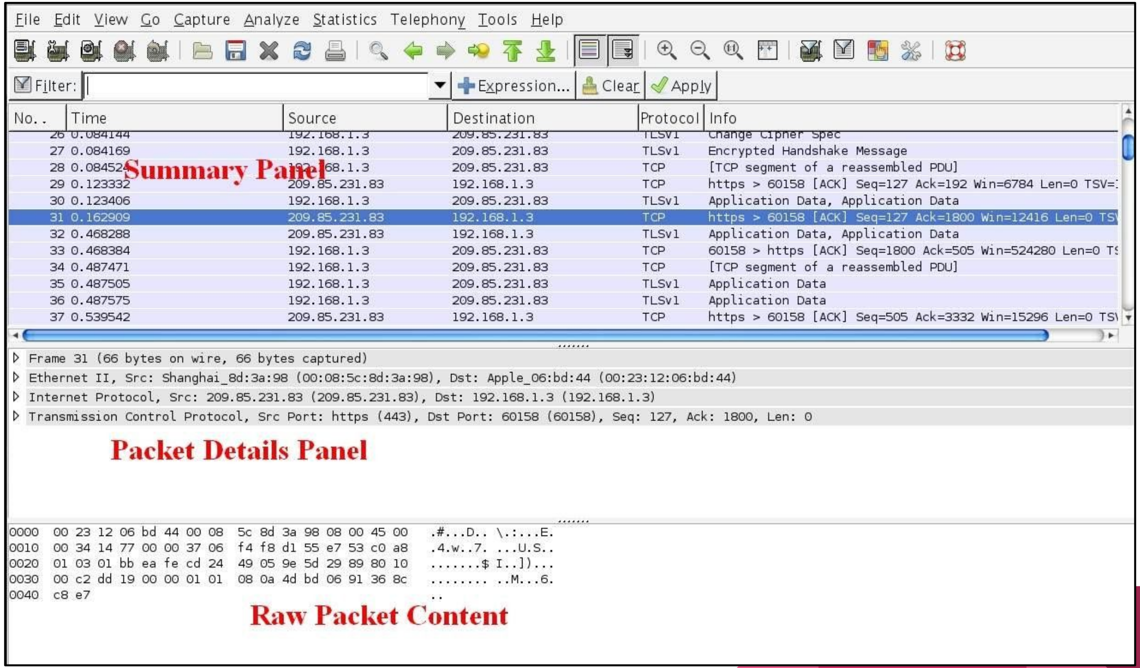
```
$ sudo tcpdump -i bridge0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on bridge0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:54:05.688236 IP 192.168.1.160.52564 > qj-in-f95.1e100.net.https: Flags [S], seq 721296916, win 29200, opti
ons [mss 1460,sackOK,TS val 19007048 ecr 0,nop,wscale 7], length 0
22:54:05.688993 IP 192.168.1.160.35622 > google-public-dns-a.google.com.domain: 59741+ PTR? 95.206.194.173.in
-addr.arpa. (45)
22:54:05.714013 IP qj-in-f95.1e100.net.https > 192.168.1.160.52564: Flags [S.], seq 1887192380, ack 721296917
, win 42540, options [mss 1430,sackOK,TS val 3990718448 ecr 19007048,nop,wscale 7], length 0
22:54:05.714082 IP 192.168.1.160.52564 > qj-in-f95.1e100.net.https: Flags [.], ack 1, win 229, options [nop,n
op,TS val 19007054 ecr 3990718448], length 0
22:54:05.720248 IP google-public-dns-a.google.com.domain > 192.168.1.160.35622: 59741 1/0/0 PTR qj-in-f95.1e1
00.net. (78)
22:54:05.720538 IP 192.168.1.160.56087 > google-public-dns-a.google.com.domain: 10609+ PTR? 160.1.168.192.in-
addr.arpa. (44)
22:54:05.730487 IP google-public-dns-a.google.com.domain > 192.168.1.160.56087: 10609 NXDomain 0/0/0 (44)
22:54:05.730776 IP 192.168.1.160.50135 > google-public-dns-a.google.com.domain: 33637+ PTR? 8.8.8.8.in-addr.a
rpa. (38)
22:54:05.765121 IP 192.168.1.160.52564 > qj-in-f95.1e100.net.https: Flags [P.], seq 296:422, ack 3591, win 29
6, options [nop,nop,TS val 19007067 ecr 3990718496], length 126
22:54:05.790913 IP qj-in-f95.1e100.net.https > 192.168.1.160.52564: Flags [P.], seq 3591:3821, ack 422, win 3
41, options [nop,nop,TS val 3990718526 ecr 19007067], length 230
22:54:05.791515 IP 192.168.1.160.52564 > qj-in-f95.1e100.net.https: Flags [P.], seq 422:741, ack 3821, win 31
8, options [nop,nop,TS val 19007074 ecr 3990718526], length 319
22:54:05.859433 IP qj-in-f95.1e100.net.https > 192.168.1.160.52564: Flags [.], ack 741, win 350, options [nop
,nop,TS val 3990718593 ecr 19007074], length 0
```

Wireshark

- Has the same functionality as Tcpdump but with a nice GUI.
- Also includes sorting and filtering features.
- Best part is you can color code it too!

Reading Wireshark Output

- The output of a packet capture tells us:
  - Source
  - Destination
  - Protocol
  - Length in bytes
  - Additional packet info



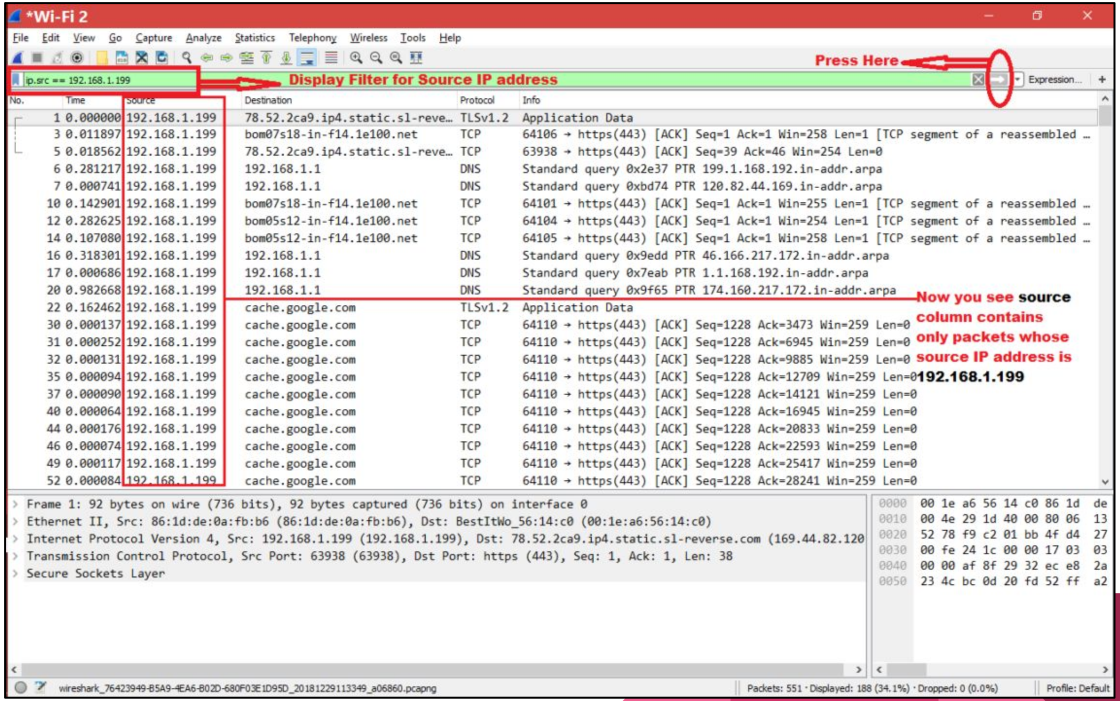
```
No. .  Time          Source          Destination     Protocol  Info
  26 0.084144      192.168.1.3     209.85.231.83   TLSv1     Change Cipher Spec
  27 0.084169      192.168.1.3     209.85.231.83   TLSv1     Encrypted Handshake Message
  28 0.084524      192.168.1.3     209.85.231.83   TCP       [TCP segment of a reassembled PDU]
  29 0.123332      209.85.231.83   192.168.1.3     TCP       https > 60158 [ACK] Seq=127 Ack=192 Win=6784 Len=0 TSV=
  30 0.123406      192.168.1.3     209.85.231.83   TLSv1     Application Data, Application Data
  31 0.162909      209.85.231.83   192.168.1.3     TCP       https > 60158 [ACK] Seq=127 Ack=1800 Win=12416 Len=0 TSV
  32 0.468288      209.85.231.83   192.168.1.3     TLSv1     Application Data, Application Data
  33 0.468384      192.168.1.3     209.85.231.83   TCP       60158 > https [ACK] Seq=1800 Ack=505 Win=524280 Len=0 TS
  34 0.487471      192.168.1.3     209.85.231.83   TCP       [TCP segment of a reassembled PDU]
  35 0.487505      192.168.1.3     209.85.231.83   TLSv1     Application Data
  36 0.487575      192.168.1.3     209.85.231.83   TLSv1     Application Data
  37 0.539542      209.85.231.83   192.168.1.3     TCP       https > 60158 [ACK] Seq=505 Ack=3332 Win=15296 Len=0 TSV
```

```
▷ Frame 31 (66 bytes on wire, 66 bytes captured)
▷ Ethernet II, Src: Shanghai_8d:3a:98 (00:08:5c:8d:3a:98), Dst: Apple_06:bd:44 (00:23:12:06:bd:44)
▷ Internet Protocol, Src: 209.85.231.83 (209.85.231.83), Dst: 192.168.1.3 (192.168.1.3)
▷ Transmission Control Protocol, Src Port: https (443), Dst Port: 60158 (60158), Seq: 127, Ack: 1800, Len: 0
```

```
0000  00 23 12 06 bd 44 00 08  5c 8d 3a 98 08 00 45 00   .#...D.. \.:...E.
0010  00 34 14 77 00 00 37 06  f4 f8 d1 55 e7 53 c0 a8   .4.w..7. ...U.S..
0020  01 03 01 bb ea fe cd 24  49 05 9e 5d 29 89 80 10   .......$ I..])...
0030  00 c2 dd 19 00 00 01 01  08 0a 4d bd 06 91 36 8c   ........ ..M...6.
0040  c8 e7                                              ..
```

Wireshark Filters

- These are your best friends!
- Saves time and saves you from a huge headache.
- Capture Filter: Determines what wireshark will capture.
- Display Filter: Filters the results of the capture.


*Wi-Fi 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

ip.src == 192.168.1.199

Display Filter for Source IP address

Press Here

```
No.  Time     Source         Destination                     Protocol Info
   1 0.000000 192.168.1.199  78.52.2ca9.ip4.static.sl-reve…  TLSv1.2  Application Data
   3 0.011897 192.168.1.199  bom07s18-in-f14.1e100.net       TCP      64106 → https(443) [ACK] Seq=1 Ack=1 Win=258 Len=1 [TCP segment of a reassembled …
   5 0.018562 192.168.1.199  78.52.2ca9.ip4.static.sl-reve…  TCP      63938 → https(443) [ACK] Seq=39 Ack=46 Win=254 Len=0
   6 0.281217 192.168.1.199  192.168.1.1                     DNS      Standard query 0x2e37 PTR 199.1.168.192.in-addr.arpa
   7 0.000741 192.168.1.199  192.168.1.1                     DNS      Standard query 0xbd74 PTR 120.82.44.169.in-addr.arpa
  10 0.142901 192.168.1.199  bom07s18-in-f14.1e100.net       TCP      64101 → https(443) [ACK] Seq=1 Ack=1 Win=255 Len=1 [TCP segment of a reassembled …
  12 0.282625 192.168.1.199  bom05s12-in-f14.1e100.net       TCP      64104 → https(443) [ACK] Seq=1 Ack=1 Win=254 Len=1 [TCP segment of a reassembled …
  14 0.107080 192.168.1.199  bom05s12-in-f14.1e100.net       TCP      64105 → https(443) [ACK] Seq=1 Ack=1 Win=258 Len=1 [TCP segment of a reassembled …
  16 0.318301 192.168.1.199  192.168.1.1                     DNS      Standard query 0x9edd PTR 46.166.217.172.in-addr.arpa
  17 0.000686 192.168.1.199  192.168.1.1                     DNS      Standard query 0x7eab PTR 1.1.168.192.in-addr.arpa
  20 0.982668 192.168.1.199  192.168.1.1                     DNS      Standard query 0x9f65 PTR 174.160.217.172.in-addr.arpa
  22 0.162462 192.168.1.199  cache.google.com                TLSv1.2  Application Data
  30 0.000137 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=3473 Win=259 Len=0
  31 0.000252 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=6945 Win=259 Len=0
  32 0.000131 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=9885 Win=259 Len=0
  35 0.000094 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=12709 Win=259 Len=0
  37 0.000090 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=14121 Win=259 Len=0
  40 0.000064 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=16945 Win=259 Len=0
  44 0.000176 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=20833 Win=259 Len=0
  46 0.000074 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=22593 Win=259 Len=0
  49 0.000117 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=25417 Win=259 Len=0
  52 0.000084 192.168.1.199  cache.google.com                TCP      64110 → https(443) [ACK] Seq=1228 Ack=28241 Win=259 Len=0
```

Now you see **source** column contains only packets whose source IP address is **192.168.1.199**

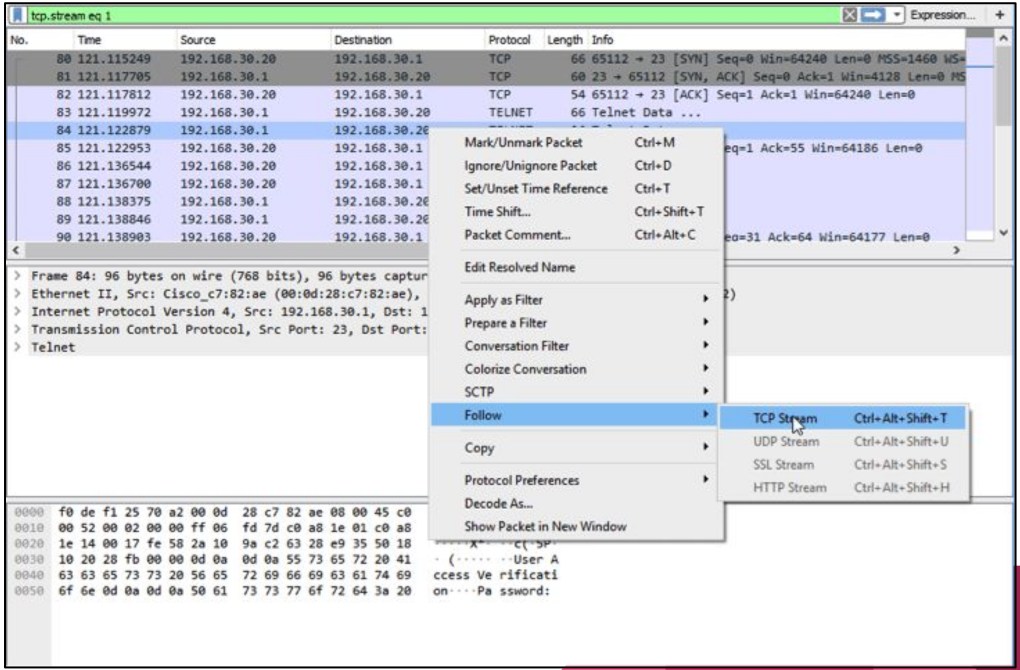
```
> Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
> Ethernet II, Src: 86:1d:de:0a:fb:b6 (86:1d:de:0a:fb:b6), Dst: BestItWo_56:14:c0 (00:1e:a6:56:14:c0)
> Internet Protocol Version 4, Src: 192.168.1.199 (192.168.1.199), Dst: 78.52.2ca9.ip4.static.sl-reverse.com (169.44.82.120
> Transmission Control Protocol, Src Port: 63938 (63938), Dst Port: https (443), Seq: 1, Ack: 1, Len: 38
> Secure Sockets Layer
```

```
0000  00 1e a6 56 14 c0 86 1d  de
0010  00 4e 29 1d 40 00 80 06  13
0020  52 78 f9 c2 01 bb 4f d4  27
0030  00 fe 24 1c 00 00 17 03  03
0040  00 00 af 8f 29 32 ec e8  2a
0050  23 4c bc 0d 20 fd 52 ff  a2
```

wireshark_76423949-B5A9-4EA6-B02D-680F03E1D95D_20181229113349_a06860.pcapng | Packets: 551 · Displayed: 188 (34.1%) · Dropped: 0 (0.0%) | Profile: Default

Using Wireshark to Analyze a Packet Capture (Pcap)

- Follow TCP and HTTP stream.
- Conversations
- These tools can be used to obtain info about who was the sender, receiver, and what was sent.
- Very good tool to graphically analyze the capture info. Includes multiple features to assist with gathering info.



tcp.stream eq 1

```
No.   Time        Source          Destination     Protocol Length Info
   80 121.115249  192.168.30.20   192.168.30.1    TCP          66 65112 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=
   81 121.117705  192.168.30.1    192.168.30.20   TCP          60 23 → 65112 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MS
   82 121.117812  192.168.30.20   192.168.30.1    TCP          54 65112 → 23 [ACK] Seq=1 Ack=1 Win=64240 Len=0
   83 121.119972  192.168.30.1    192.168.30.20   TELNET       66 Telnet Data ...
   84 121.122879  192.168.30.1    192.168.30.2
   85 121.122953  192.168.30.20   192.168.30.1                    eq=1 Ack=55 Win=64186 Len=0
   86 121.136544  192.168.30.20   192.168.30.1
   87 121.136700  192.168.30.20   192.168.30.1
   88 121.138375  192.168.30.1    192.168.30.2
   89 121.138846  192.168.30.1    192.168.30.2
   90 121.138903  192.168.30.20   192.168.30.1                    eq=31 Ack=64 Win=64177 Len=0
```

```
> Frame 84: 96 bytes on wire (768 bits), 96 bytes captur
> Ethernet II, Src: Cisco_c7:82:ae (00:0d:28:c7:82:ae),                2)
> Internet Protocol Version 4, Src: 192.168.30.1, Dst: 1
> Transmission Control Protocol, Src Port: 23, Dst Port:
> Telnet
```

```
0000  f0 de f1 25 70 a2 00 0d  28 c7 82 ae 08 00 45 c0
0010  00 52 00 02 00 00 ff 06  fd 7d c0 a8 1e 01 c0 a8
0020  1e 14 00 17 fe 58 2a 10  9a c2 63 28 e9 35 50 18
0030  10 20 28 fb 00 00 0d 0a  0d 0a 55 73 65 72 20 41   · (····· ··User A
0040  63 63 65 73 73 20 56 65  72 69 66 69 63 61 74 69   ccess Ve rificati
0050  6f 6e 0d 0a 0d 0a 50 61  73 73 77 6f 72 64 3a 20   on····Pa ssword:
```

Expression...
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
SSL Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H

Snort

- Snorts main functionality is as an IDS/IPS.
- Snort has three modes:
  - **Sniffer Mode**
    - The program will read network packets and display them on the console.
  - **Packet Logger mode**
    - In packet logger mode, the program will save the capture data.
  - **Network Intrusion Detection System Mode**
    - In intrusion detection mode, the program will monitor network traffic and analyze it against a rule set defined by the user and perform a specific action based on what is identified.
- The Packet Logger mode allows for pcap analysis.

Zeek (Bro)

- Main functionality is to analyze network traffic in the form of a pcap.
- Can be used as an IDS but with additional live analysis of network events.
- Produces several logs such as:
    - Conn.log
    - Dns.log
    - Ftp.log
    - Http.log
    - Files.log
    - Ssh.log
    - Weird.log

VirusTotal and Google

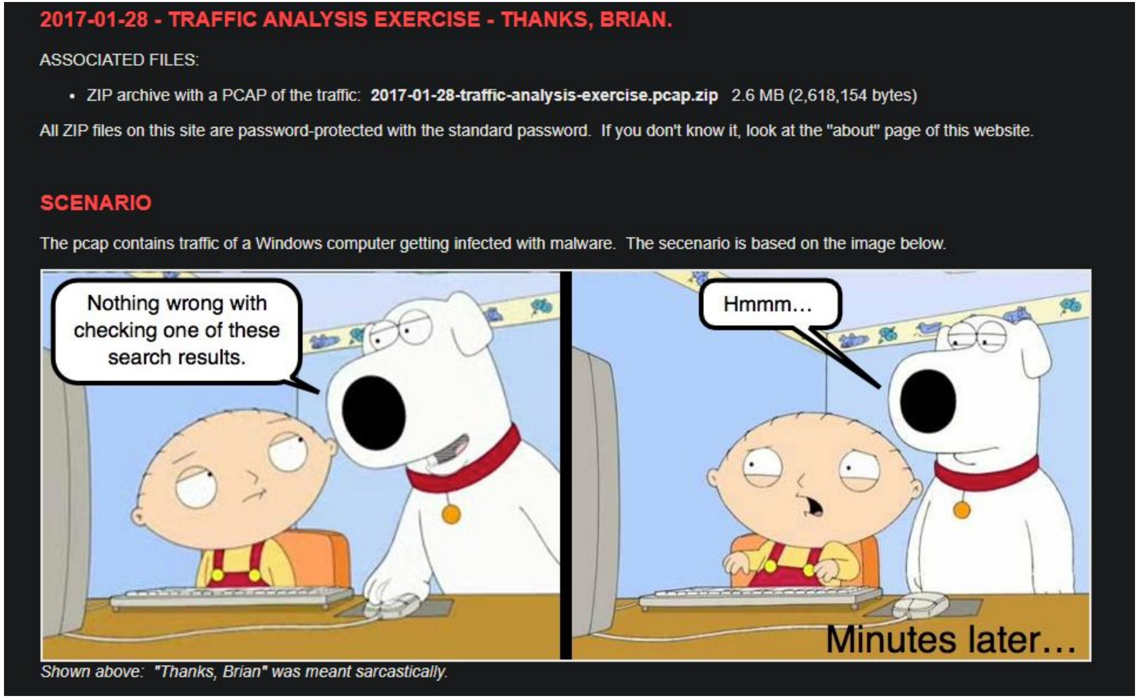
- Believe it or not, but sites like VirusTotal and Google can be a huge asset in packet analysis.
- Once you have found something that looks suspicious, you can verify it with VirusTotal or Google to see if it is malicious or not.
- This includes websites, files, IPs, etc.

Demo

- Now we will capture live HTTP traffic using Wireshark to help give you a taste of what to expect for the HW.

HW

- **PLEASE START EARLY!!**
- Analyze the provided pcap to answer these questions:
  - Who was infected?
  - How were they infected?
  - What were they infected with?
  - How could this be prevented from happening again in the future?



2017-01-28 - TRAFFIC ANALYSIS EXERCISE - THANKS, BRIAN.

ASSOCIATED FILES:

- ZIP archive with a PCAP of the traffic: **2017-01-28-traffic-analysis-exercise.pcap.zip** 2.6 MB (2,618,154 bytes)

All ZIP files on this site are password-protected with the standard password. If you don't know it, look at the "about" page of this website.

SCENARIO

The pcap contains traffic of a Windows computer getting infected with malware. The secenario is based on the image below.

Shown above: "Thanks, Brian" was meant sarcastically.