CS 798: Digital Forensics and Incident Response

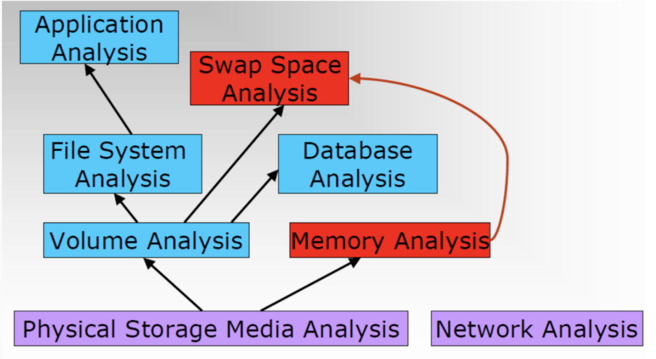
Lecture 12 - Network Traffic Analysis

Diogo Barradas

Winter 2025

University of Waterloo

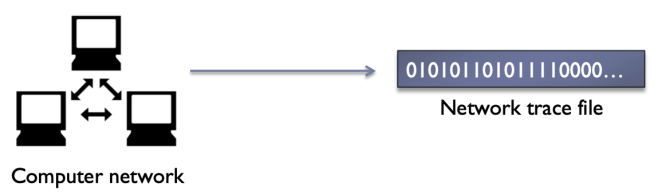
Evidence in network traffic



Source: „File System Forensic Analysis", Brian Carrier

- What are the major sources of evidences from the network?
- Which techniques can be used to extract and analyze them?

Network traffic analysis



- Allow us to collect information from network traces
- A network trace is a linearized bit-copy of collected data exchanged over the network
- Packet analysis, flow analysis, protocol analysis

Why would we want to analyze traffic?

- Detection of potential / and actual attacks
  - e.g., port scans, denial of service attacks
- Reverse engineering of communication protocols
  - e.g., analysis of botnet chatter, or proprietary protocols
- Inspect the contents of communications
  - e.g., intercept message exchanges, carve out file transmissions
- Detect payload patterns
  - e.g., website fingerprinting, DRM-protected content, Tor traffic

Outline

1. Packet analysis techniques

2. Flow analysis techniques

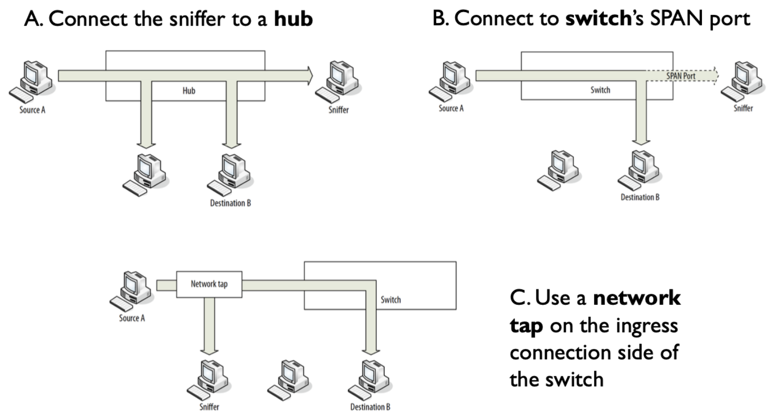
3. Protocol analysis techniques

Packet analysis techniques

Packet sniffers and network traces

- Packet sniffing is the act of looking at packets as computers pass them over networks
- Packet sniffing is performed using packet sniffers
  - These programs are designed to capture raw data as it crosses the network and translate it into a human readable format for analysis
  - Can be used to capture only relevant packets
- Packet sniffers range from simple, command-line programs, like `tcpdump`, to complex programs with GUI

Where packet sniffers are usually placed



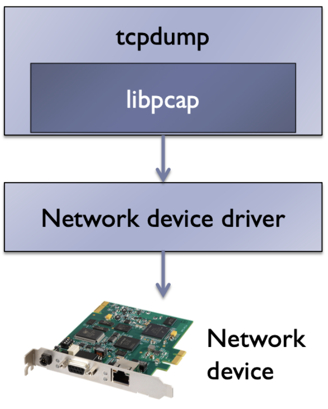
A. Connect the sniffer to a **hub**

B. Connect to **switch**’s SPAN port

C. Use a **network tap** on the ingress connection side of the switch

Packet sniffers: `tcpdump`

- `tcpdump` is the grandfather of open source packet sniffers
- Uses `libpcap`, which contains a set of system- independent functions for packet capture and network analysis
    - Also used by Wireshark



Packet analysis

- Involves the examination of contents and/or metadata of one or more packets
- Conducted to identify packets of interest and develop a strategy for flow analysis or content reconstruction

- There are many tools available for packet analysis
  - e.g., Wireshark (and `tshark`), `ngrep`, ...

Main packet analysis techniques

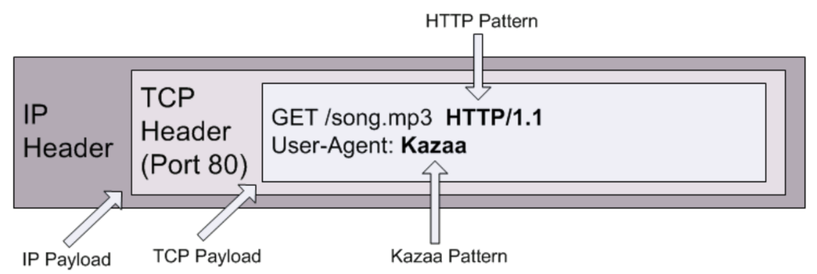
- **Parsing protocol fields**
  - Extract the contents of protocol fields within packets of interest (e.g., obtain TCP fields of packets)
- **Packet filtering**
  - Separate packets based on the values of fields in protocol metadata (e.g., filter interesting conversation snippet)
- **Pattern matching**
  - Identify packets of interest by matching specific values within the packet capture (e.g., keyword search)

Pattern matching

- Search the packet capture for patterns of interest
- Pattern examples:
  1. String match
  2. Source/destination match
  3. Numerical properties

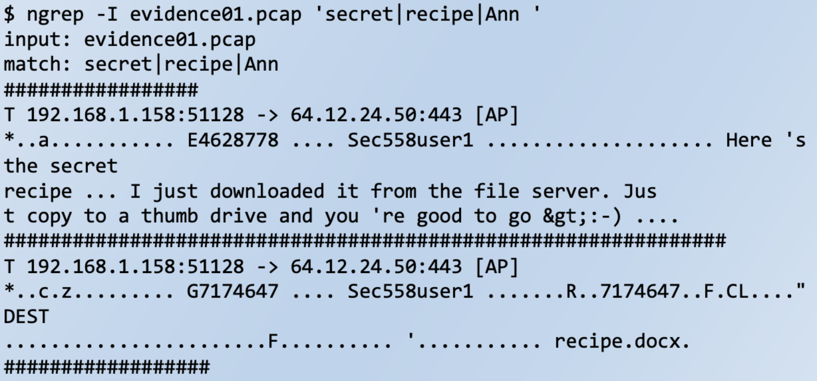
Analysis by string matching

- String search in packet headers
  1. Many applications have pure textual identifiers
  2. Very easy if in a specific location within a packet
  3. Uniqueness not always guaranteed



Analysis by string matching

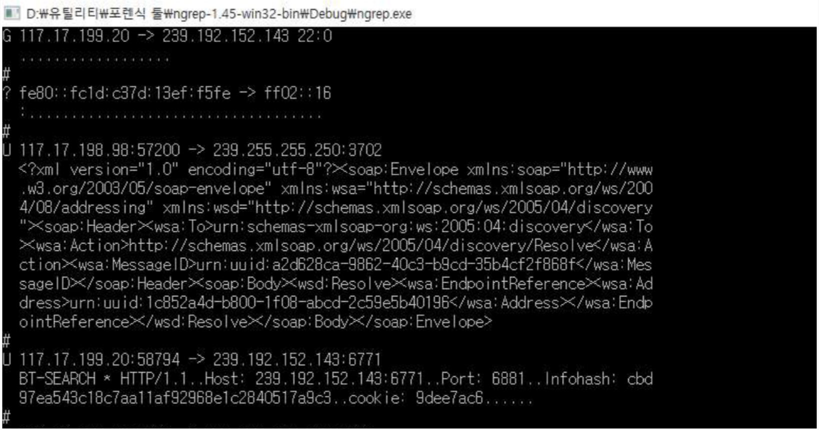
- String search in packet payload
    1. Example: Packets containing string from some list



```
$ ngrep -I evidence01.pcap 'secret|recipe|Ann '
input: evidence01.pcap
match: secret|recipe|Ann
#################
T 192.168.1.158:51128 -> 64.12.24.50:443 [AP]
*..a........... E4628778 .... Sec558user1 .................... Here 's
the secret
recipe ... I just downloaded it from the file server. Jus
t copy to a thumb drive and you 're good to go &gt;:-) ....
###############################################################
T 192.168.1.158:51128 -> 64.12.24.50:443 [AP]
*..c.z......... G7174647 .... Sec558user1 .......R..7174647..F.CL...."
DEST
.......................F.......... '........... recipe.docx.
##################
```

Analysis by source/destination

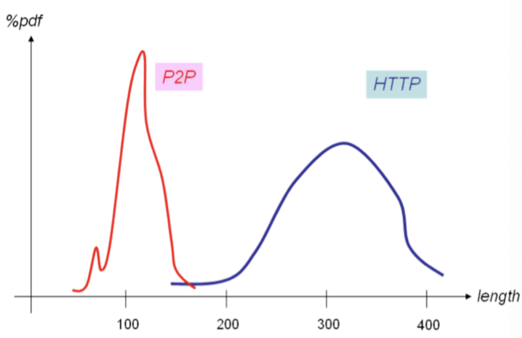
- Example: Packets from 117.17.199.20 to 239.192.152.143



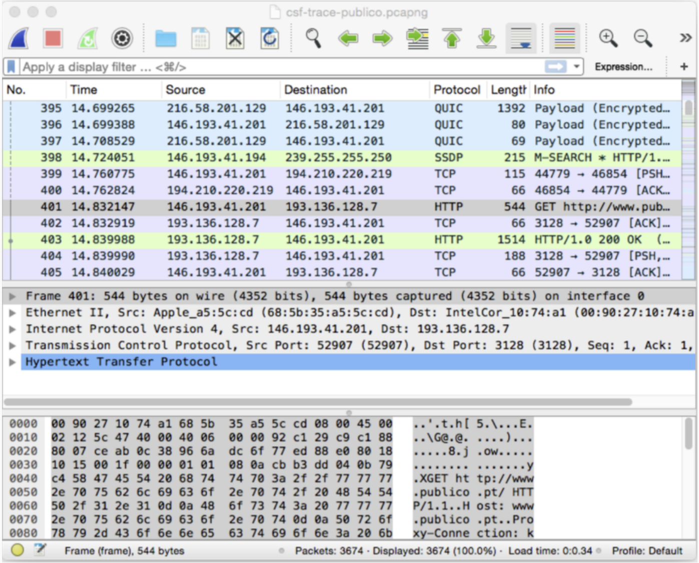
```
G 117.17.199.20 -> 239.192.152.143 22:0
  ..................
#
? fe80::fc1d:c37d:13ef:f5fe -> ff02::16
  :...................................
#
U 117.17.198.98:57200 -> 239.255.255.250:3702
  <?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://www
  .w3.org/2003/05/soap-envelope" xmlns:wsa="http://schemas.xmlsoap.org/ws/200
  4/08/addressing" xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery
  "><soap:Header><wsa:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</wsa:To
  ><wsa:Action>http://schemas.xmlsoap.org/ws/2005/04/discovery/Resolve</wsa:A
  ction><wsa:MessageID>urn:uuid:a2d628ca-9862-40c3-b9cd-35b4cf2f868f</wsa:Mes
  sageID></soap:Header><soap:Body><wsd:Resolve><wsa:EndpointReference><wsa:Ad
  dress>urn:uuid:1c852a4d-b800-1f08-abcd-2c59e5b40196</wsa:Address></wsa:Endp
  ointReference></wsd:Resolve></soap:Body></soap:Envelope>
#
U 117.17.199.20:58794 -> 239.192.152.143:6771
  BT-SEARCH * HTTP/1.1..Host: 239.192.152.143:6771..Port: 6881..Infohash: cbd
  97ea543c18c7aa11af92968e1c2840517a9c3..cookie: 9dee7ac6......
#
```

Analysis by numerical properties

- One can look beyond content and focus on metadata:
  1. Packet size
  2. Payload/message length
  3. Position within packet
- Statistics: on average payload size is between X to Y
  1. Very effective analysis when applications use encryption



Use Case #1: Inspecting individual packets


csf-trace-publico.pcapng
Apply a display filter … <⌘/>
Expression…
+
Frame (frame), 544 bytes
Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.34
Profile: Default

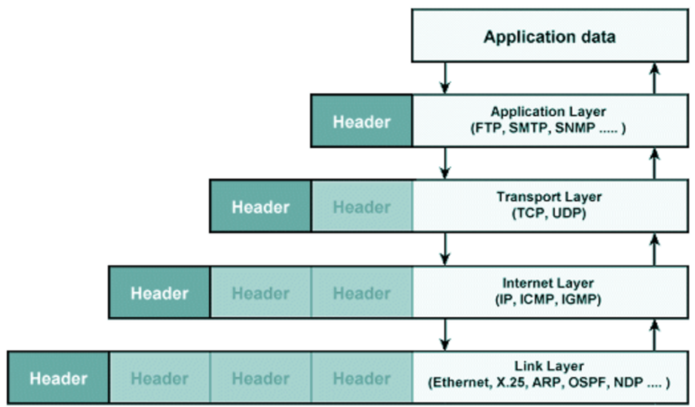
No.	Time	Source	Destination	Protocol	Length	Info
395	14.699265	216.58.201.129	146.193.41.201	QUIC	1392	Payload (Encrypted…
396	14.699388	146.193.41.201	216.58.201.129	QUIC	80	Payload (Encrypted…
397	14.708529	216.58.201.129	146.193.41.201	QUIC	69	Payload (Encrypted…
398	14.724051	146.193.41.194	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.…
399	14.760775	146.193.41.201	194.210.220.219	TCP	115	44779 → 46854 [PSH…
400	14.762824	194.210.220.219	146.193.41.201	TCP	66	46854 → 44779 [ACK…
401	14.832147	146.193.41.201	193.136.128.7	HTTP	544	GET http://www.pub…
402	14.832919	193.136.128.7	146.193.41.201	TCP	66	3128 → 52907 [ACK]…
403	14.839988	193.136.128.7	146.193.41.201	HTTP	1514	HTTP/1.0 200 OK (…
404	14.839990	193.136.128.7	146.193.41.201	TCP	188	3128 → 52907 [PSH,…
405	14.840029	146.193.41.201	193.136.128.7	TCP	66	52907 → 3128 [ACK]…

```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▶ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74:a
▶ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
▶ Transmission Control Protocol, Src Port: 52907 (52907), Dst Port: 3128 (3128), Seq: 1, Ack: 1,
▶ Hypertext Transfer Protocol
```

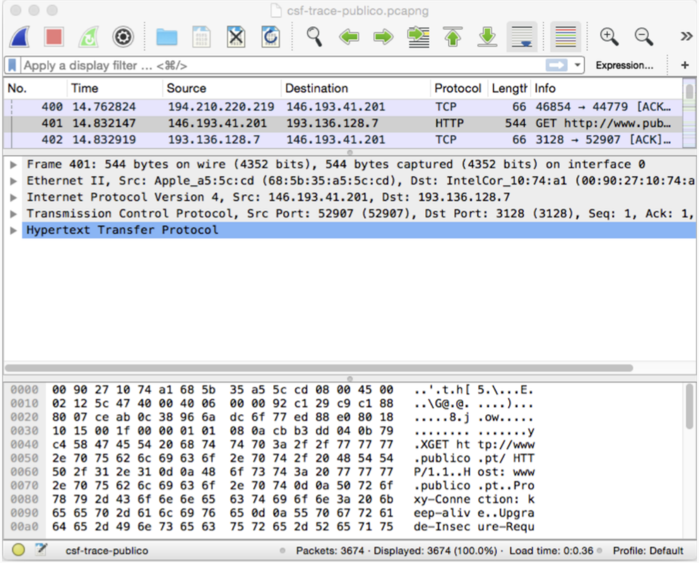
```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   .....8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
0070  2e 70 75 62 6c 69 63 6f  2e 70 74 0d 0a 50 72 6f   .publico .pt..Pro
0080  78 79 2d 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   xy-Conne ction: k
```

Packet network layers

- Packets are encoded according to network layers
    - Each layer plays a role in abstracting out details of lower levels



Wireshark lets us navigate across each layer



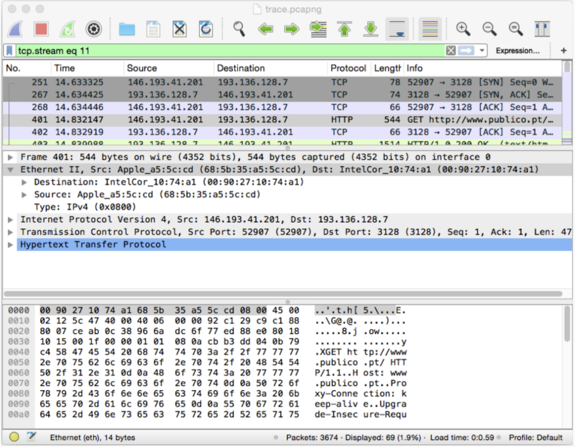
```
No.   Time       Source           Destination     Protocol Length Info
  400 14.762824  194.210.220.219  146.193.41.201  TCP        66  46854 → 44779 [ACK…
  401 14.832147  146.193.41.201   193.136.128.7   HTTP      544  GET http://www.pub…
  402 14.832919  193.136.128.7    146.193.41.201  TCP        66  3128 → 52907 [ACK]…
```

```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▶ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74:a
▶ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
▶ Transmission Control Protocol, Src Port: 52907 (52907), Dst Port: 3128 (3128), Seq: 1, Ack: 1,
▶ Hypertext Transfer Protocol
```

```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   .....8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
0070  2e 70 75 62 6c 69 63 6f  2e 70 74 0d 0a 50 72 6f   .publico .pt..Pro
0080  78 79 2d 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   xy-Conne ction: k
0090  65 65 70 2d 61 6c 69 76  65 0d 0a 55 70 67 72 61   eep-aliv e..Upgra
00a0  64 65 2d 49 6e 73 65 63  75 72 65 2d 52 65 71 75   de-Insec ure-Requ
```

csf-trace-publico · Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.36 · Profile: Default

Ethernet frame of an IP packet



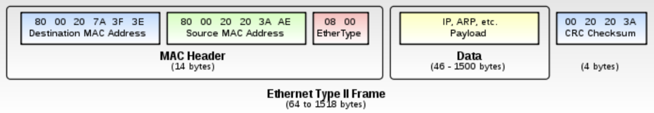
```
tcp.stream eq 11
```

```
No.   Time        Source          Destination     Protocol Length Info
  251 14.633325   146.193.41.201  193.136.128.7   TCP        78 52907 → 3128 [SYN] Seq=0 W…
  267 14.634425   193.136.128.7   146.193.41.201  TCP        74 3128 → 52907 [SYN, ACK] Se…
  268 14.634446   146.193.41.201  193.136.128.7   TCP        66 52907 → 3128 [ACK] Seq=1 A…
  401 14.832147   146.193.41.201  193.136.128.7   HTTP      544 GET http://www.publico.pt/…
  402 14.832919   193.136.128.7   146.193.41.201  TCP        66 3128 → 52907 [ACK] Seq=1 A…
  403 14.839088   193.136.128.7   146.193.41.201  HTTP     1514 HTTP/1.0 200 OK  (text/htm
```

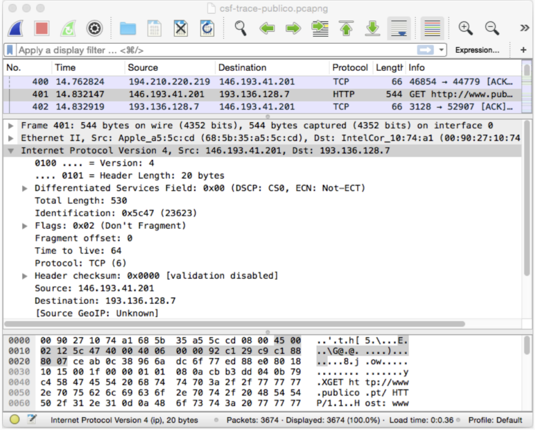
```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▼ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74:a1)
   ▶ Destination: IntelCor_10:74:a1 (00:90:27:10:74:a1)
   ▶ Source: Apple_a5:5c:cd (68:5b:35:a5:5c:cd)
     Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
▶ Transmission Control Protocol, Src Port: 52907 (52907), Dst Port: 3128 (3128), Seq: 1, Ack: 1, Len: 47
▶ Hypertext Transfer Protocol
```

```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   .....8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
0070  2e 70 75 62 6c 69 63 6f  2e 70 74 0d 0a 50 72 6f   .publico .pt..Pro
0080  78 79 2d 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   xy-Conne ction: k
0090  65 65 70 2d 61 6c 69 76  65 0d 0a 55 70 67 72 61   eep-aliv e..Upgra
00a0  64 65 2d 49 6e 73 65 63  75 72 65 2d 52 65 71 75   de-Insec ure-Requ
```

Ethernet (eth), 14 bytes · Packets: 3674 · Displayed: 69 (1.9%) · Load time: 0:0.59 · Profile: Default



The Internet Protocol

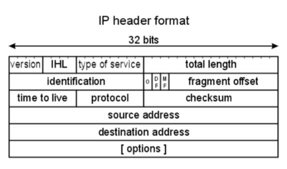


No.	Time	Source	Destination	Protocol	Length	Info
400	14.762824	194.210.220.219	146.193.41.201	TCP	66	46854 → 44779 [ACK…
401	14.832147	146.193.41.201	193.136.128.7	HTTP	544	GET http://www.pub…
402	14.832919	193.136.128.7	146.193.41.201	TCP	66	3128 → 52907 [ACK]…

```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▶ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74
▼ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 530
    Identification: 0x5c47 (23623)
  ▶ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▶ Header checksum: 0x0000 [validation disabled]
    Source: 146.193.41.201
    Destination: 193.136.128.7
    [Source GeoIP: Unknown]
```

```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   ......8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
```

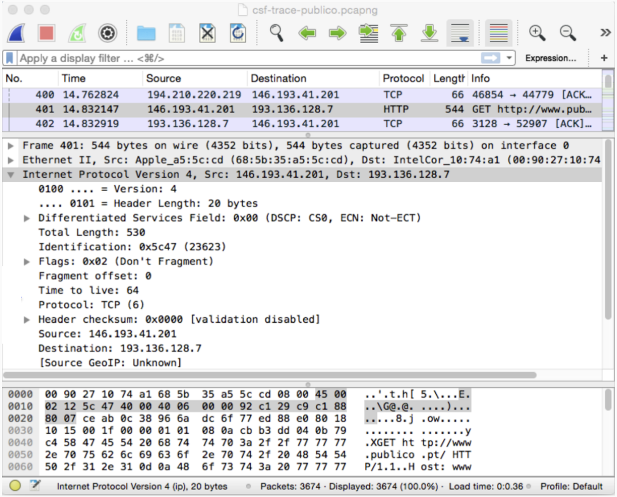
Internet Protocol Version 4 (ip), 20 bytes · Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.36 · Profile: Default


IP header format

32 bits

<table>
<tr><td colspan="4">version</td><td colspan="4">IHL</td><td colspan="8">type of service</td><td colspan="16">total length</td></tr>
<tr><td colspan="16">identification</td><td>0</td><td>D
F</td><td>M
F</td><td colspan="13">fragment offset</td></tr>
<tr><td colspan="8">time to live</td><td colspan="8">protocol</td><td colspan="16">checksum</td></tr>
<tr><td colspan="32">source address</td></tr>
<tr><td colspan="32">destination address</td></tr>
<tr><td colspan="32">[options]</td></tr>
</table>

Parsing the IP packet payload



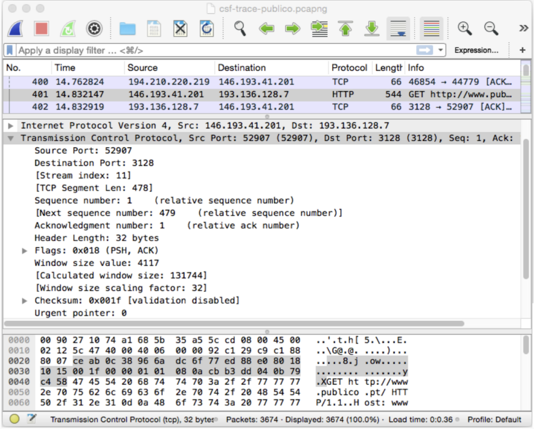
```
No.   Time       Source            Destination      Protocol  Length  Info
  400 14.762824  194.210.220.219   146.193.41.201   TCP          66   46854 → 44779 [ACK…
  401 14.832147  146.193.41.201    193.136.128.7    HTTP        544   GET http://www.pub…
  402 14.832919  193.136.128.7     146.193.41.201   TCP          66   3128 → 52907 [ACK]…
```

```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▶ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74
▼ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 530
    Identification: 0x5c47 (23623)
  ▶ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▶ Header checksum: 0x0000 [validation disabled]
    Source: 146.193.41.201
    Destination: 193.136.128.7
    [Source GeoIP: Unknown]
```

```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   .....8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
```

Internet Protocol Version 4 (ip), 20 bytes · Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.36 · Profile: Default

The TCP Protocol

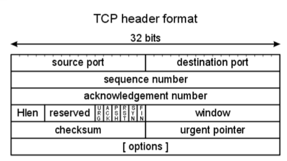


No.	Time	Source	Destination	Protocol	Lengtl	Info
400	14.762824	194.210.220.219	146.193.41.201	TCP	66	46854 → 44779 [ACK…
401	14.832147	146.193.41.201	193.136.128.7	HTTP	544	GET http://www.pub…
402	14.832919	193.136.128.7	146.193.41.201	TCP	66	3128 → 52907 [ACK]…

```
▶ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
▼ Transmission Control Protocol, Src Port: 52907 (52907), Dst Port: 3128 (3128), Seq: 1, Ack:
      Source Port: 52907
      Destination Port: 3128
      [Stream index: 11]
      [TCP Segment Len: 478]
      Sequence number: 1    (relative sequence number)
      [Next sequence number: 479    (relative sequence number)]
      Acknowledgment number: 1    (relative ack number)
      Header Length: 32 bytes
   ▶ Flags: 0x018 (PSH, ACK)
      Window size value: 4117
      [Calculated window size: 131744]
      [Window size scaling factor: 32]
   ▶ Checksum: 0x001f [validation disabled]
      Urgent pointer: 0
```

```
0000  00 90 27 10 74 a1 68 5b  35 a5 5c cd 08 00 45 00   ..'.t.h[ 5.\...E.
0010  02 12 5c 47 40 00 40 06  00 00 92 c1 29 c9 c1 88   ..\G@.@. ....)...
0020  80 07 ce ab 0c 38 96 6a  dc 6f 77 ed 88 e0 80 18   .....8.j .ow.....
0030  10 15 00 1f 00 00 01 01  08 0a cb b3 dd 04 0b 79   ........ .......y
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
```

Transmission Control Protocol (tcp), 32 bytes · Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.36 · Profile: Default

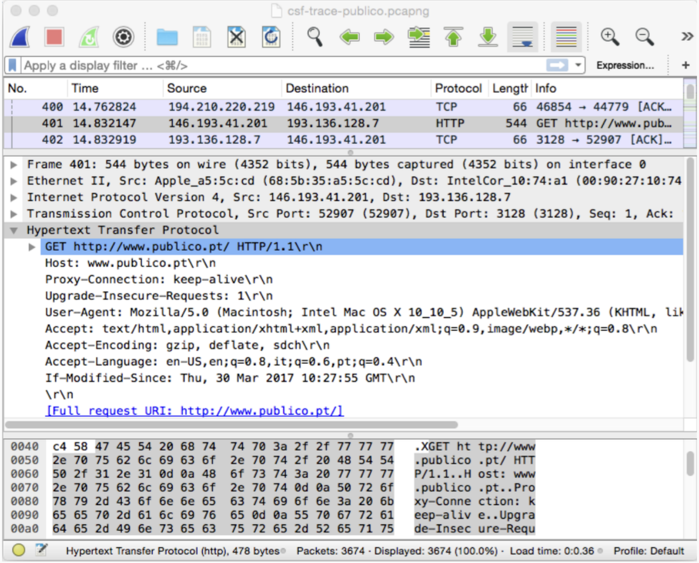


TCP header format

32 bits

source port				destination port
sequence number				
acknowledgement number				
Hlen	reserved	URG ACK PSH RST SYN FIN		window
checksum				urgent pointer
[options]				

HTTP Request



No.	Time	Source	Destination	Protocol	Length	Info
400	14.762824	194.210.220.219	146.193.41.201	TCP	66	46854 → 44779 [ACK…
401	14.832147	146.193.41.201	193.136.128.7	HTTP	544	GET http://www.pub…
402	14.832919	193.136.128.7	146.193.41.201	TCP	66	3128 → 52907 [ACK]…

```
▶ Frame 401: 544 bytes on wire (4352 bits), 544 bytes captured (4352 bits) on interface 0
▶ Ethernet II, Src: Apple_a5:5c:cd (68:5b:35:a5:5c:cd), Dst: IntelCor_10:74:a1 (00:90:27:10:74
▶ Internet Protocol Version 4, Src: 146.193.41.201, Dst: 193.136.128.7
▶ Transmission Control Protocol, Src Port: 52907 (52907), Dst Port: 3128 (3128), Seq: 1, Ack:
▼ Hypertext Transfer Protocol
  ▶ GET http://www.publico.pt/ HTTP/1.1\r\n
    Host: www.publico.pt\r\n
    Proxy-Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, lik
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate, sdch\r\n
    Accept-Language: en-US,en;q=0.8,it;q=0.6,pt;q=0.4\r\n
    If-Modified-Since: Thu, 30 Mar 2017 10:27:55 GMT\r\n
    \r\n
    [Full request URI: http://www.publico.pt/]
```

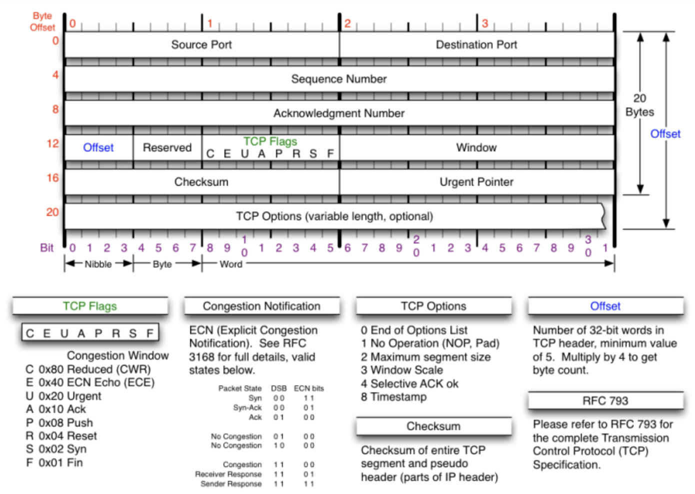
```
0040  c4 58 47 45 54 20 68 74  74 70 3a 2f 2f 77 77 77   .XGET ht tp://www
0050  2e 70 75 62 6c 69 63 6f  2e 70 74 2f 20 48 54 54   .publico .pt/ HTT
0060  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 77 77 77   P/1.1..H ost: www
0070  2e 70 75 62 6c 69 63 6f  2e 70 74 0d 0a 50 72 6f   .publico .pt..Pro
0080  78 79 2d 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   xy-Conne ction: k
0090  65 65 70 2d 61 6c 69 76  65 0d 0a 55 70 67 72 61   eep-aliv e..Upgra
00a0  64 65 2d 49 6e 73 65 63  75 72 65 2d 52 65 71 75   de-Insec ure-Requ
```

Hypertext Transfer Protocol (http), 478 bytes · Packets: 3674 · Displayed: 3674 (100.0%) · Load time: 0:0.36 · Profile: Default

Use Case #2: Detection of port scans

- Port scan attacks: aim to detect whether or not there is a service listening on a specific port
- Main techniques
    - SYN scan
    - ACK scan
    - UDP scan

TCP review: TCP header



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

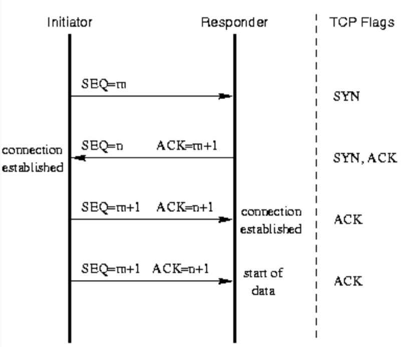
Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

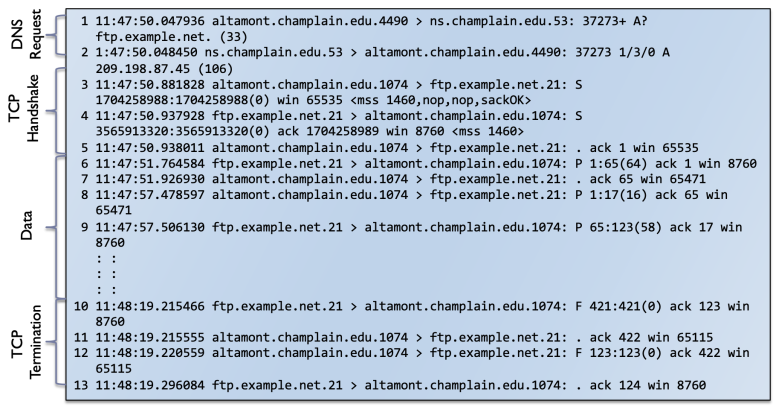
Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

TCP review: TCP 3-way handshake

- Client sends a SYN to the server. Client sets the segment's sequence number to rand value `m`
- Server replies with a SYN-ACK. The ack number is set to `m`+1, and the sequence number that the server chooses for the packet is another random number `n`
- Client sends an ACK back to the server



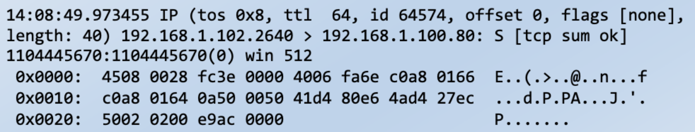
TCP review: Connection setup



```
 1 11:47:50.047936 altamont.champlain.edu.4490 > ns.champlain.edu.53: 37273+ A?
   ftp.example.net. (33)
 2 1:47:50.048450 ns.champlain.edu.53 > altamont.champlain.edu.4490: 37273 1/3/0 A
   209.198.87.45 (106)
 3 11:47:50.881828 altamont.champlain.edu.1074 > ftp.example.net.21: S
   1704258988:1704258988(0) win 65535 <mss 1460,nop,nop,sackOK>
 4 11:47:50.937928 ftp.example.net.21 > altamont.champlain.edu.1074: S
   3565913320:3565913320(0) ack 1704258989 win 8760 <mss 1460>
 5 11:47:50.938011 altamont.champlain.edu.1074 > ftp.example.net.21: . ack 1 win 65535
 6 11:47:51.764584 ftp.example.net.21 > altamont.champlain.edu.1074: P 1:65(64) ack 1 win 8760
 7 11:47:51.926930 altamont.champlain.edu.1074 > ftp.example.net.21: . ack 65 win 65471
 8 11:47:57.478597 altamont.champlain.edu.1074 > ftp.example.net.21: P 1:17(16) ack 65 win
   65471
 9 11:47:57.506130 ftp.example.net.21 > altamont.champlain.edu.1074: P 65:123(58) ack 17 win
   8760
   : :
   : :
   : :
10 11:48:19.215466 ftp.example.net.21 > altamont.champlain.edu.1074: F 421:421(0) ack 123 win
   8760
11 11:48:19.215555 altamont.champlain.edu.1074 > ftp.example.net.21: . ack 422 win 65115
12 11:48:19.220559 altamont.champlain.edu.1074 > ftp.example.net.21: F 123:123(0) ack 422 win
   65115
13 11:48:19.296084 ftp.example.net.21 > altamont.champlain.edu.1074: . ack 124 win 8760
```

SYN scan

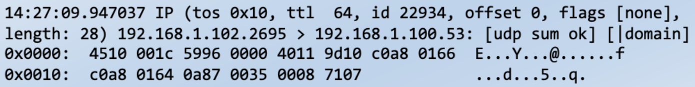
- One of the most common scans out there today
  - If there is a service, then that would elicit a SYN/ACK
  - If the result is RST/ACK then there is no service listening
- SYN scan to 192.168.1.100 on port 80



```
14:08:49.973455 IP (tos 0x8, ttl  64, id 64574, offset 0, flags [none],
length: 40) 192.168.1.102.2640 > 192.168.1.100.80: S [tcp sum ok]
1104445670:1104445670(0) win 512
 0x0000:  4508 0028 fc3e 0000 4006 fa6e c0a8 0166  E..(.>..@..n...f
 0x0010:  c0a8 0164 0a50 0050 41d4 80e6 4ad4 27ec  ...d.P.PA...J.'.
 0x0020:  5002 0200 e9ac 0000                      P.......
```

UDP scan

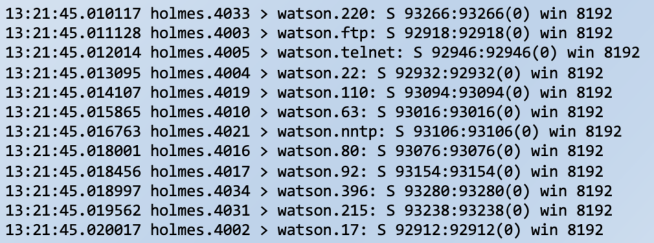
- Useful for discovering UDP based services such as DNS
  - If computer has a service listening you will get nothing back
  - Otherwise, you will get an ICMP port unreachable message
- UDP scan to 192.168.1.100 on port 53



```
14:27:09.947037 IP (tos 0x10, ttl  64, id 22934, offset 0, flags [none],
length: 28) 192.168.1.102.2695 > 192.168.1.100.53: [udp sum ok] [|domain]
0x0000:  4510 001c 5996 0000 4011 9d10 c0a8 0166  E...Y...@......f
0x0010:  c0a8 0164 0a87 0035 0008 7107            ...d...5..q.
```

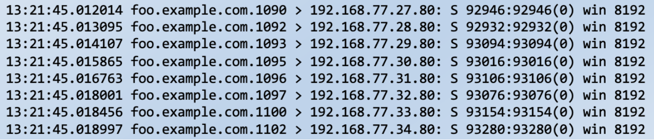
Analysis by source / destination

- Port scan using TCP ports from a user-defined list
  - From host `holmes` to host `watson` (responses not shown)



```
13:21:45.010117 holmes.4033 > watson.220: S 93266:93266(0) win 8192
13:21:45.011128 holmes.4003 > watson.ftp: S 92918:92918(0) win 8192
13:21:45.012014 holmes.4005 > watson.telnet: S 92946:92946(0) win 8192
13:21:45.013095 holmes.4004 > watson.22: S 92932:92932(0) win 8192
13:21:45.014107 holmes.4019 > watson.110: S 93094:93094(0) win 8192
13:21:45.015865 holmes.4010 > watson.63: S 93016:93016(0) win 8192
13:21:45.016763 holmes.4021 > watson.nntp: S 93106:93106(0) win 8192
13:21:45.018001 holmes.4016 > watson.80: S 93076:93076(0) win 8192
13:21:45.018456 holmes.4017 > watson.92: S 93154:93154(0) win 8192
13:21:45.018997 holmes.4034 > watson.396: S 93280:93280(0) win 8192
13:21:45.019562 holmes.4031 > watson.215: S 93238:93238(0) win 8192
13:21:45.020017 holmes.4002 > watson.17: S 92912:92912(0) win 8192
```

- Site scan for any Web servers (listening on port 80)
  - Host `foo.example.net` hits hosts on 192.168.77.0 subnet



```
13:21:45.012014 foo.example.com.1090 > 192.168.77.27.80: S 92946:92946(0) win 8192
13:21:45.013095 foo.example.com.1092 > 192.168.77.28.80: S 92932:92932(0) win 8192
13:21:45.014107 foo.example.com.1093 > 192.168.77.29.80: S 93094:93094(0) win 8192
13:21:45.015865 foo.example.com.1095 > 192.168.77.30.80: S 93016:93016(0) win 8192
13:21:45.016763 foo.example.com.1096 > 192.168.77.31.80: S 93106:93106(0) win 8192
13:21:45.018001 foo.example.com.1097 > 192.168.77.32.80: S 93076:93076(0) win 8192
13:21:45.018456 foo.example.com.1100 > 192.168.77.33.80: S 93154:93154(0) win 8192
13:21:45.018997 foo.example.com.1102 > 192.168.77.34.80: S 93280:93280(0) win 8192
```

Use case #3: Detection of DDoS attacks

- UDP flood
  - Flood random ports on a remote host with UDP packets
- ICMP flood
  - Overwhelm the target with ICMP Echo Request (ping) packets
- Smurf attack
  - Send ICMP packets with the victim's spoofed source IP
- SYN flood
  - Send multiple SYN requests, but either does not respond to SYN-ACK response, or sends SYN requests from a spoofed IP
- Amplification
  - e.g., exploit publically-accessible NTP servers to overwhelm the targeted server with UDP traffic
- HTTP flood
  - Seemingly-legitimate HTTP GET or POST requests to attack a web server or application

Flow analysis techniques

Go with the flow

- In RFC 3679, a flow is defined as:
  - “a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow”
- A flow can consist of all packets in a specific transport connection or a media stream
  - But, not necessarily 1:1 mapped to a transport connection
  - Can be constructed upon other L4 protocols, including UDP

Flow analysis

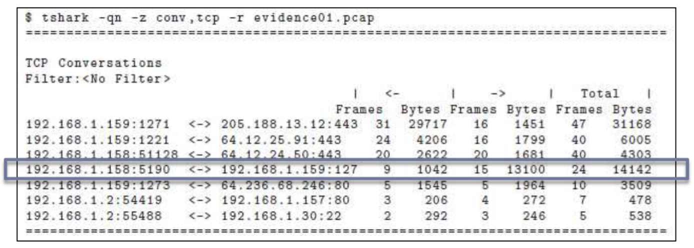
- Flow analysis consists in examination of sequences of related packets (i.e., flows)
- Conducted to identify traffic patterns, isolate suspicious activity, analyze higher-layer protocols, or extract data
- Examples of flow analysis tools: Wireshark, `tcpflow`, `pcapcat`

Summary of flow analysis techniques

1. List flows
    - List all flows within a packet capture, or only specific flows based on their characteristics
2. Export a flow
    - Isolate a flow, or multiple flows, and store the flow(s) of interest to disk for further analysis
3. File and data carving
    - Extract files or other data of interest from reassembled flow

1. List flows

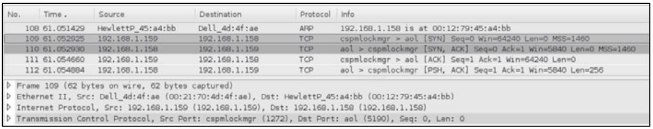
- Listing conversations (TCP streams) using `tshark`
    - 192.168.1.158 on TCP port 5190
    - host involved in the conversation was 192.168.1.159
    - 1,042 bytes were transferred



```
$ tshark -qn -z conv,tcp -r evidence01.pcap
================================================================================

TCP Conversations
Filter:<No Filter>
                                         |    <-      |    ->      |   Total    |
                                       Frames  Bytes Frames Bytes Frames Bytes
192.168.1.159:1271  <-> 205.188.13.12:443  31   29717   16   1451   47   31168
192.168.1.159:1221  <-> 64.12.25.91:443    24    4206   16   1799   40    6005
192.168.1.158:51128 <-> 64.12.24.50:443    20    2622   20   1681   40    4303
192.168.1.158:5190  <-> 192.168.1.159:127   9    1042   15  13100   24   14142
192.168.1.159:1273  <-> 64.236.68.246:80    5    1545    5   1964   10    3509
192.168.1.2:54419   <-> 192.168.1.157:80    3     206    4    272    7     478
192.168.1.2:55488   <-> 192.168.1.30:22     2     292    3    246    5     538
================================================================================
```

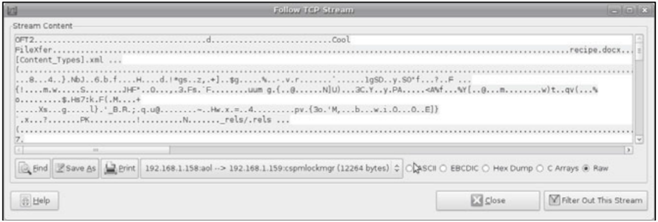
2. Export a flow



```
No.   Time .        Source              Destination       Protocol  Info
  108 61.051429  HewlettP_45:a4:bb   Dell_4d:4f:ae       ARP       192.168.1.158 is at 00:12:79:45:a4:bb
  109 61.052925  192.168.1.159       192.168.1.158       TCP       cspmlockmgr > aol [SYN] Seq=0 Win=64240 Len=0 MSS=1460
  110 61.052930  192.168.1.158       192.168.1.159       TCP       aol > cspmlockmgr [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
  111 61.054660  192.168.1.159       192.168.1.158       TCP       cspmlockmgr > aol [ACK] Seq=1 Ack=1 Win=64240 Len=0
  112 61.054884  192.168.1.158       192.168.1.159       TCP       aol > cspmlockmgr [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=256

▷ Frame 109 (62 bytes on wire, 62 bytes captured)
▷ Ethernet II, Src: Dell_4d:4f:ae (00:21:70:4d:4f:ae), Dst: HewlettP_45:a4:bb (00:12:79:45:a4:bb)
▷ Internet Protocol, Src: 192.168.1.159 (192.168.1.159), Dst: 192.168.1.158 (192.168.1.158)
▷ Transmission Control Protocol, Src Port: cspmlockmgr (1272), Dst Port: aol (5190), Seq: 0, Len: 0
```

- Frame #109 is part of the TCP stream of interest
    - 192.168.1.158 -192.168.1.159 on TCP port 5190



Stream Content

```
OFT2.....................................d............................Cool
FileXfer...........................................................................................................recipe.docx...
[Content_Types].xml ...
(...............................................................................................................................
...8...4..}.NbJ..6.b.f....H....d.!*gs..z,.+]..$g.....%..-.v.r........`......1gSD..y.SO"f...?..F ...
{!....m.w.....S........JHF"..O...,.3.Fs.`F........uum g.{..@......N]U)...3C.Y..y.PA.....<A%f...%Y[..@...m........w)t..qv(...%
o.........$.Hs7:k.F(.M....+
.....Xs...g.....l}.'_B.R.;.q.u@........~..Hw.x.=..4.........pv.{3o.'M,...b...w.i.O...O..E]}
`.x...?.......PK..........!.........N......._rels/.rels ...
(...............................................................................................................................
7.
```

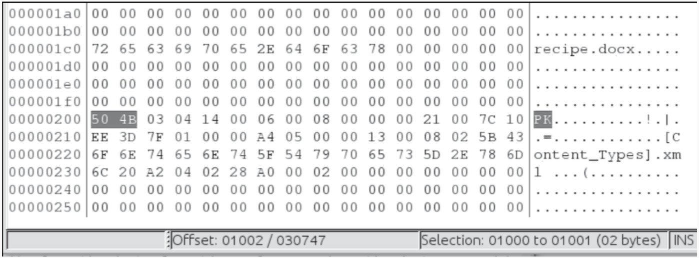
192.168.1.158:aol --> 192.168.1.159:cspmlockmgr (12264 bytes)

○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

- "Follow TCP Stream" function to isolate TCP stream, and select direction which appears to contain OFT2 file

3. File / data carving

- Carve a file out of the captured network traffic
  - A `.docx` file has the “magic number” `0x50 0x4B`



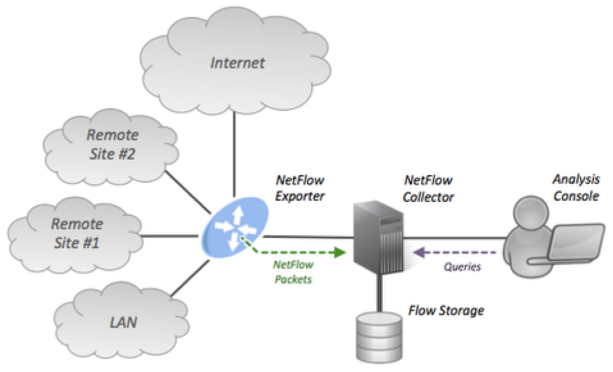
```
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001c0 72 65 63 69 70 65 2E 64 6F 63 78 00 00 00 00 00 recipe.docx.....
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000200 50 4B 03 04 14 00 06 00 08 00 00 00 21 00 7C 10 PK..........!.|.
00000210 EE 3D 7F 01 00 00 A4 05 00 00 13 00 08 02 5B 43 .=............[C
00000220 6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D ontent_Types].xm
00000230 6C 20 A2 04 02 28 A0 00 02 00 00 00 00 00 00 00 l ...(..........
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

Offset: 01002 / 030747    Selection: 01000 to 01001 (02 bytes)    INS
```

- To determine the file ending, add the file size number determined from the OFT header (omitted here)

Netflow logs

- Technology by Cisco that collects and categorizes IP traffic as it passes through the supported network devices
- Built into the supported devices, NetFlow does not collect the entire payload of the network packets
    - It creates a cache on the router for each new flow



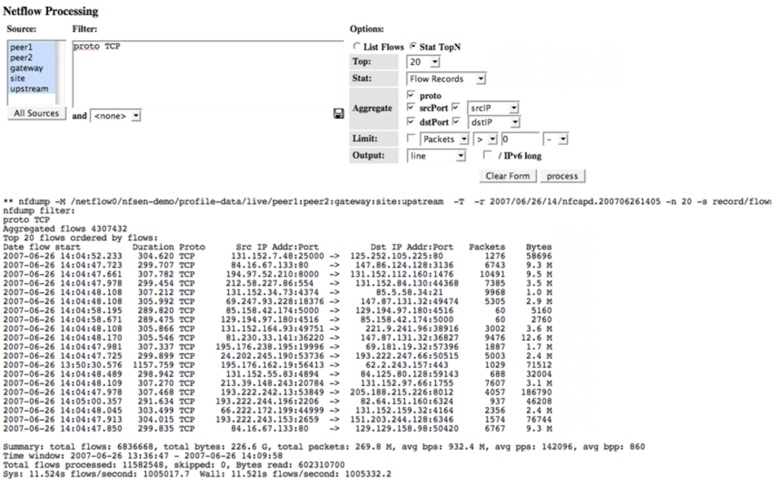
Netflow logs and its benefits to forensics

- NetFlows logs flows, not TCP connections!
- For each TCP connection, NetFlow records two flows
- Benefits:
  - Greatly reduce the amount of data needed to be analyzed
  - Simpler to identify any suspicious traffic for future investigation

Which packets belong to individual flows?

- As IP packets come into a supported device interface, NetFlow scans them for the following seven fields:
    1. Source IP address
    2. Destination IP address
    3. Source port number
    4. Destination port number
    5. IP protocol
    6. Type-of-service (ToS) byte
    7. Input logical interface
- If these fields match an existing flow the byte count for the flow entry is incremented within the device cache
    - Else the packet is part of new flow

NetFlow listing example



Netflow Processing

Source: peer1, peer2, gateway, site, upstream

Filter: proto TCP

and <none>

Options:

○ List Flows ● Stat TopN

Top: 20

Stat: Flow Records

Aggregate ☑ proto
☑ srcPort ☑ srcIP
☑ dstPort ☑ dstIP

Limit: □ Packets > 0 -

Output: line □ / IPv6 long

```
** nfdump -M /netflow0/nfsen-demo/profile-data/live/peer1:peer2:gateway:site:upstream  -T  -r 2007/06/26/14/nfcapd.200706261405 -n 20 -s record/flow
nfdump filter:
proto TCP
Aggregated flows 4307432
Top 20 flows ordered by flows:
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes
2007-06-26 14:04:52.233   304.620 TCP       131.152.7.48:25000 -> 125.252.105.225:80       1276    58696
2007-06-26 14:04:47.723   299.707 TCP        84.16.67.133:80    ->   147.86.124.128:3136     6743    9.3 M
2007-06-26 14:04:47.661   307.782 TCP      194.97.52.210:8000  -> 131.152.112.160:1476    10491    9.5 M
2007-06-26 14:04:47.978   299.454 TCP      212.58.227.86:554   ->  131.152.84.130:44368    7385    3.5 M
2007-06-26 14:04:48.108   307.212 TCP      131.152.34.73:4374  ->      85.5.58.34:21        9968    1.0 M
2007-06-26 14:04:48.108   305.992 TCP      69.247.93.228:18376 ->   147.87.131.32:49474    5305    2.9 M
2007-06-26 14:04:58.195   289.820 TCP      85.158.42.174:5000  ->  129.194.97.180:4516       60     5160
2007-06-26 14:04:58.671   289.475 TCP     129.194.97.180:4516  ->   85.158.42.174:5000       60     2760
2007-06-26 14:04:48.108   305.866 TCP     131.152.164.93:49751 ->     221.9.241.96:38916    3002    3.6 M
2007-06-26 14:04:48.170   305.546 TCP      81.230.33.141:36220 ->   147.87.131.32:36827    9476   12.6 M
2007-06-26 14:04:47.981   307.337 TCP    195.176.238.195:19996 ->     69.181.19.32:57396    1887    1.7 M
2007-06-26 14:04:47.725   299.899 TCP     24.202.245.190:53736 ->  193.222.247.66:50515    5003    2.4 M
2007-06-26 13:50:30.576  1157.759 TCP     195.176.162.19:56413 ->     62.2.243.157:443      1029    71512
2007-06-26 14:04:48.489   298.942 TCP      131.152.55.83:4894  ->   84.125.80.128:59143     688    32004
2007-06-26 14:04:48.109   307.270 TCP     213.39.148.243:20784 ->   131.152.97.66:1755     7607    3.1 M
2007-06-26 14:04:47.978   307.468 TCP     193.222.242.13:53849 -> 205.188.215.226:8012     4057   186790
2007-06-26 14:05:00.357   291.634 TCP    193.222.244.196:2206  ->   82.64.151.160:6324      937    46208
2007-06-26 14:04:48.045   303.499 TCP     66.222.172.199:44999 ->  131.152.159.32:4164     2356    2.4 M
2007-06-26 14:04:47.913   304.015 TCP    193.222.243.153:2659  -> 151.203.244.128:6346     1574    76744
2007-06-26 14:04:47.850   299.835 TCP        84.16.67.133:80    ->  129.129.158.98:50420    6767    9.3 M

Summary: total flows: 6836668, total bytes: 226.6 G, total packets: 269.8 M, avg bps: 932.4 M, avg pps: 142096, avg bpp: 860
Time window: 2007-06-26 13:36:47 - 2007-06-26 14:09:58
Total flows processed: 11582548, skipped: 0, Bytes read: 602310700
Sys: 11.524s flows/second: 1005017.7  Wall: 11.521s flows/second: 1005332.2
```

Protocol analysis techniques

Protocol analysis

- Aim to understand how a particular communication protocol works, what it's used for, how to identify it, how to dissect it
- Many protocols are deliberately kept secret:
    - By their inventors to protect intellectual property, keep out competition, or security and covert communications
    - By attackers to bypass IDS (Intrusion Detection Systems) firewall rules, smuggle data in strange places, and generally create mayhem

Where to get information on protocols

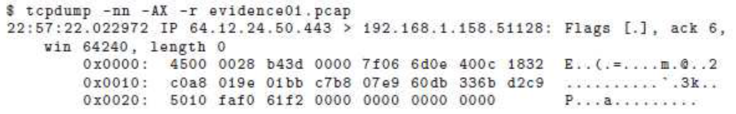
- IETF Request for Comments (RFC)
  - Perhaps the most well known public repository of documented protocols
  - RFC: any thought, suggestion, etc. related to the Internet network
- Other standard bodies
  - Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA)
  - International Organization for Standardization (ISO)
- Vendors
  - Cisco: RFC2784
  - Microsoft: communications protocols used by Window server, clients
- Researchers
  - Russian researcher Alexandr Shutko has published his "Unofficial" OSCAR (ICQ v7/v8/v9) protocol documentation

Protocol identification techniques

1. Search for common binary/hexadecimal/ASCII values that are typically associated with a specific protocol
2. Leverage information in the encapsulating protocol
3. Leverage the TCP/UDP port number, many of which are associated with standard default services
4. Analyze the function of the source or destination server (specified by IP address or hostname)
5. Test for the presence of recognizable protocol structures
6. Check metadata for matches with known protocols

1. Search values associated w/ specific protocol

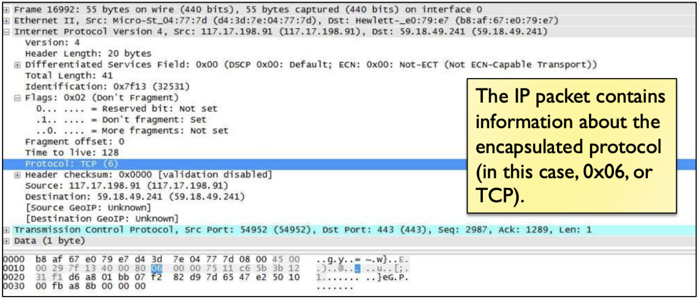
- Search for common binary/hexadecimal/ASCII values that are typically associated with a specific protocol
  - Most protocols contain sequences of bits
  - Present in packets associated with that protocol, in predictable places
  - Hexadecimal sequence `0x45 0x00` often marks the start of an IPv4 packet



```
$ tcpdump -nn -AX -r evidence01.pcap
22:57:22.022972 IP 64.12.24.50.443 > 192.168.1.158.51128: Flags [.], ack 6,
    win 64240, length 0
        0x0000:  4500 0028 b43d 0000 7f06 6d0e 400c 1832  E..(.=....m.@..2
        0x0010:  c0a8 019e 01bb c7b8 07e9 60db 336b d2c9  ..........`.3k..
        0x0020:  5010 faf0 61f2 0000 0000 0000 0000       P...a.........
```

2. Information in the encapsulating protocol

- Leverage information in the encapsulating protocol
  - Protocols often contain info that indicates the type of encapsulated protocol
  - OSI model, lower-layer protocol fields typically indicate the higher-layer protocol that may be encapsulated, to facilitate proper processing



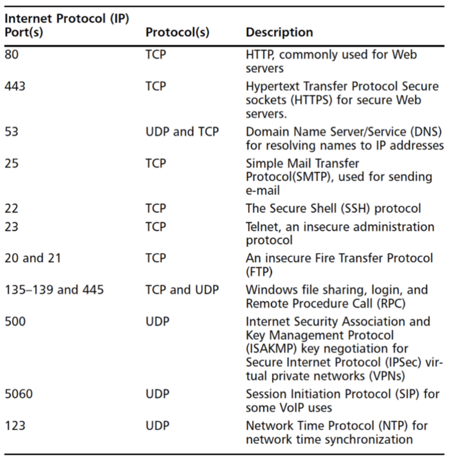
```
⊞ Frame 16992: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
⊞ Ethernet II, Src: Micro-St_04:77:7d (d4:3d:7e:04:77:7d), Dst: Hewlett-_e0:79:e7 (b8:af:67:e0:79:e7)
⊟ Internet Protocol Version 4, Src: 117.17.198.91 (117.17.198.91), Dst: 59.18.49.241 (59.18.49.241)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 41
    Identification: 0x7f13 (32531)
  ⊟ Flags: 0x02 (Don't Fragment)
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
  ⊞ Header checksum: 0x0000 [validation disabled]
    Source: 117.17.198.91 (117.17.198.91)
    Destination: 59.18.49.241 (59.18.49.241)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Transmission Control Protocol, Src Port: 54952 (54952), Dst Port: 443 (443), Seq: 2987, Ack: 1289, Len: 1
⊞ Data (1 byte)

0000  b8 af 67 e0 79 e7 d4 3d  7e 04 77 7d 08 00 45 00   ..g.y..= ~.w}..E.
0010  00 29 7f 13 40 00 80 06  00 00 75 11 c6 5b 3b 12   .)..@... ..u..[;.
0020  31 f1 d6 a8 01 bb 07 f2  82 d9 7d 65 47 e2 50 10   1....... ..}eG.P.
0030  00 fb a8 8b 00 00 00                               .......
```

The IP packet contains information about the encapsulated protocol (in this case, 0x06, or TCP).

3. Leverage the TCP/UDP port number

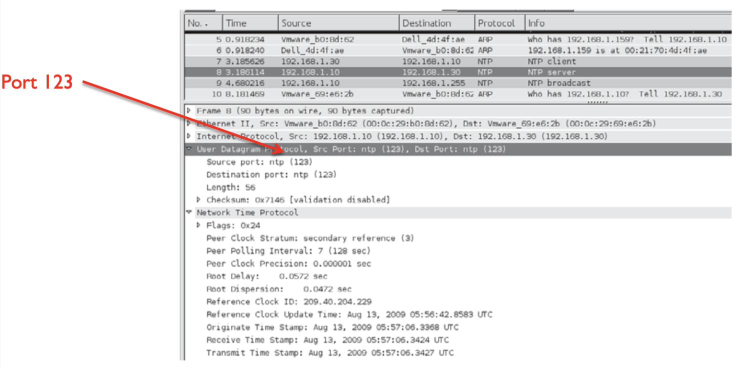
- Examine the TCP or UDP port number in use
    - Many of which are associated with standard default services



Internet Protocol (IP) Port(s)	Protocol(s)	Description
80	TCP	HTTP, commonly used for Web servers
443	TCP	Hypertext Transfer Protocol Secure sockets (HTTPS) for secure Web servers.
53	UDP and TCP	Domain Name Server/Service (DNS) for resolving names to IP addresses
25	TCP	Simple Mail Transfer Protocol(SMTP), used for sending e-mail
22	TCP	The Secure Shell (SSH) protocol
23	TCP	Telnet, an insecure administration protocol
20 and 21	TCP	An insecure Fire Transfer Protocol (FTP)
135–139 and 445	TCP and UDP	Windows file sharing, login, and Remote Procedure Call (RPC)
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) key negotiation for Secure Internet Protocol (IPSec) virtual private networks (VPNs)
5060	UDP	Session Initiation Protocol (SIP) for some VoIP uses
123	UDP	Network Time Protocol (NTP) for network time synchronization

Example using port identification

- Wireshark automatically associates the UDP port, 123, with its IANA-assigned default service, NTP



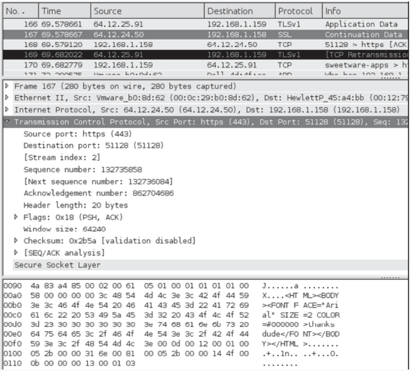
Port 123

No.	Time	Source	Destination	Protocol	Info
5	0.918234	Vmware_b0:8d:62	Dell_4d:4f:ae	ARP	Who has 192.168.1.159? Tell 192.168.1.10
6	0.918240	Dell_4d:4f:ae	Vmware_b0:8d:62	ARP	192.168.1.159 is at 00:21:70:4d:4f:ae
7	3.185626	192.168.1.30	192.168.1.10	NTP	NTP client
8	3.186114	192.168.1.10	192.168.1.30	NTP	NTP server
9	4.680216	192.168.1.10	192.168.1.255	NTP	NTP broadcast
10	8.181469	Vmware_69:e6:2b	Vmware_b0:8d:62	ARP	Who has 192.168.1.10? Tell 192.168.1.30

```
▷ Frame 8 (90 bytes on wire, 90 bytes captured)
▷ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: Vmware_69:e6:2b (00:0c:29:69:e6:2b)
▷ Internet Protocol, Src: 192.168.1.10 (192.168.1.10), Dst: 192.168.1.30 (192.168.1.30)
▽ User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)
    Source port: ntp (123)
    Destination port: ntp (123)
    Length: 56
  ▷ Checksum: 0x7146 [validation disabled]
▽ Network Time Protocol
  ▷ Flags: 0x24
    Peer Clock Stratum: secondary reference (3)
    Peer Polling Interval: 7 (128 sec)
    Peer Clock Precision: 0.000001 sec
    Root Delay:    0.0572 sec
    Root Dispersion:    0.0472 sec
    Reference Clock ID: 209.40.204.229
    Reference Clock Update Time: Aug 13, 2009 05:56:42.8583 UTC
    Originate Time Stamp: Aug 13, 2009 05:57:06.3368 UTC
    Receive Time Stamp: Aug 13, 2009 05:57:06.3424 UTC
    Transmit Time Stamp: Aug 13, 2009 05:57:06.3427 UTC
```

Limitations of port-based identification

- Servers can be configured to use nonstandard port numbers for services
    - Wireshark automatically associates TCP port 443 with its IANA-assigned default service: HTTPS
- But this is INCORRECT:
    - Packet contents not encrypted



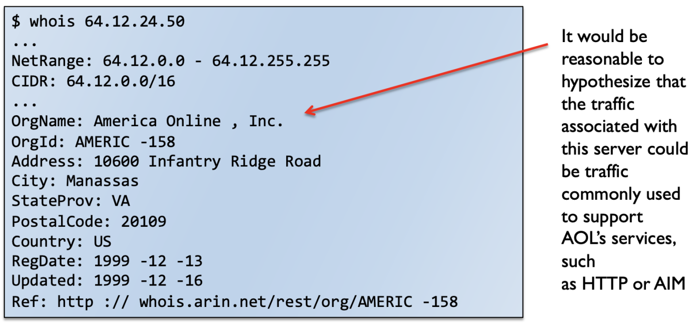
```
No. .  Time       Source          Destination     Protocol  Info
   166 69.578661  64.12.25.91     192.168.1.159   TLSv1     Application Data
   167 69.578667  64.12.24.50     192.168.1.158   SSL       Continuation Data
   168 69.579120  192.168.1.158   64.12.24.50     TCP       51128 > https [ACK
   169 69.682022  64.12.25.91     192.168.1.159   TLSv1     [TCP Retransmissio
   170 69.682779  192.168.1.159   64.12.25.91     TCP       sweetware-apps > h
```

```
▷ Frame 167 (280 bytes on wire, 280 bytes captured)
▷ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: HewlettP_45:a4:bb (00:12:79
▷ Internet Protocol, Src: 64.12.24.50 (64.12.24.50), Dst: 192.168.1.158 (192.168.1.158)
▽ Transmission Control Protocol, Src Port: https (443), Dst Port: 51128 (51128), Seq: 132
    Source port: https (443)
    Destination port: 51128 (51128)
    [Stream index: 2]
    Sequence number: 132735858
    [Next sequence number: 132736084]
    Acknowledgement number: 862704686
    Header length: 20 bytes
  ▷ Flags: 0x18 (PSH, ACK)
    Window size: 64240
  ▷ Checksum: 0x2b5a [validation disabled]
  ▷ [SEQ/ACK analysis]
  Secure Socket Layer
```

```
0090  4a 83 a4 85 00 02 00 61  05 01 00 01 01 01 01 00   J......a ........
00a0  58 00 00 00 00 3c 48 54  4d 4c 3e 3c 42 4f 44 59   X....<HT ML><BODY
00b0  3e 3c 46 4f 4e 54 20 46  41 43 45 3d 22 41 72 69   ><FONT F ACE="Ari
00c0  61 6c 22 20 53 49 5a 45  3d 32 20 43 4f 4c 4f 52   al" SIZE =2 COLOR
00d0  3d 23 30 30 30 30 30 30  3e 74 68 61 6e 6b 73 20   =#000000 >thanks
00e0  64 75 64 65 3c 2f 46 4f  4e 54 3e 3c 2f 42 4f 44   dude</FO NT></BOD
00f0  59 3e 3c 2f 48 54 4d 4c  3e 00 0d 00 12 00 01 00   Y></HTML >.......
0100  05 2b 00 00 31 6e 00 81  00 05 2b 00 00 14 4f 00   .+..1n.. ..+...O.
0110  0b 00 00 00 13 00 01 03                            ........
```

4. Analyze the function of source / destination

- Server hostnames and domains provide clues as to their functions, which can help identify likely protocols in use



```
$ whois 64.12.24.50
...
NetRange: 64.12.0.0 - 64.12.255.255
CIDR: 64.12.0.0/16
...
OrgName: America Online , Inc.
OrgId: AMERIC -158
Address: 10600 Infantry Ridge Road
City: Manassas
StateProv: VA
PostalCode: 20109
Country: US
RegDate: 1999 -12 -13
Updated: 1999 -12 -16
Ref: http :// whois.arin.net/rest/org/AMERIC -158
```

It would be reasonable to hypothesize that the traffic associated with this server could be traffic commonly used to support AOL's services, such as HTTP or AIM

Takeaways

- Traffic analysis enables forensic investigators to determine the history of events involved in network communications
- Such analysis is based on specific evidence that can can be acquired using protocol, packet, and flow analysis techniques
- It is up to the forensic investigator to select the best technique(s) to use according to the investigative scenario

Pointers

- **Textbook:**
  - Casey – Chapters 24.4, 24.5 , Luttgens – Chapter 9.4
- **Others:**
  - Introducing Network Analysis
  - Gary Kessler, On teaching TCP/IP Protocol Analysis to Computer Forensic Examiners
- **Acknowledgements:**
  - Slides adapted from Nuno Santos's Forensics Cyber-Security course at Técnico Lisbon