DIRECTDEFENSE

Introduction to Network Traffic Analysis

Stephen Deck, GSE, OSCE, CISSP

BE INFORMED. BE STRATEGIC. BE SECURE.

Overview

- Part 1: The Basics
- Part 2: Tools
- Part 3: Common Protocols
- Part 4: Scapy
- Part 5: Analysis Tips
- Part 6: Practical Exercise

Part 1 – The Basics

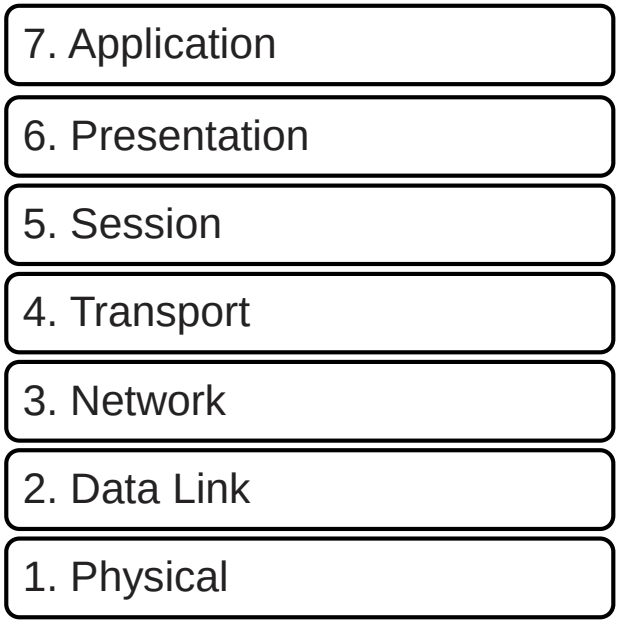
Overview
Part 1 – The Basics

- OSI & TCP/IP Model Overview
- Hexadecimal Numbers
- Binary Numbers
- Ethernet Headers
- IP Headers
- TCP Headers
- UDP Headers
- ICMP Headers

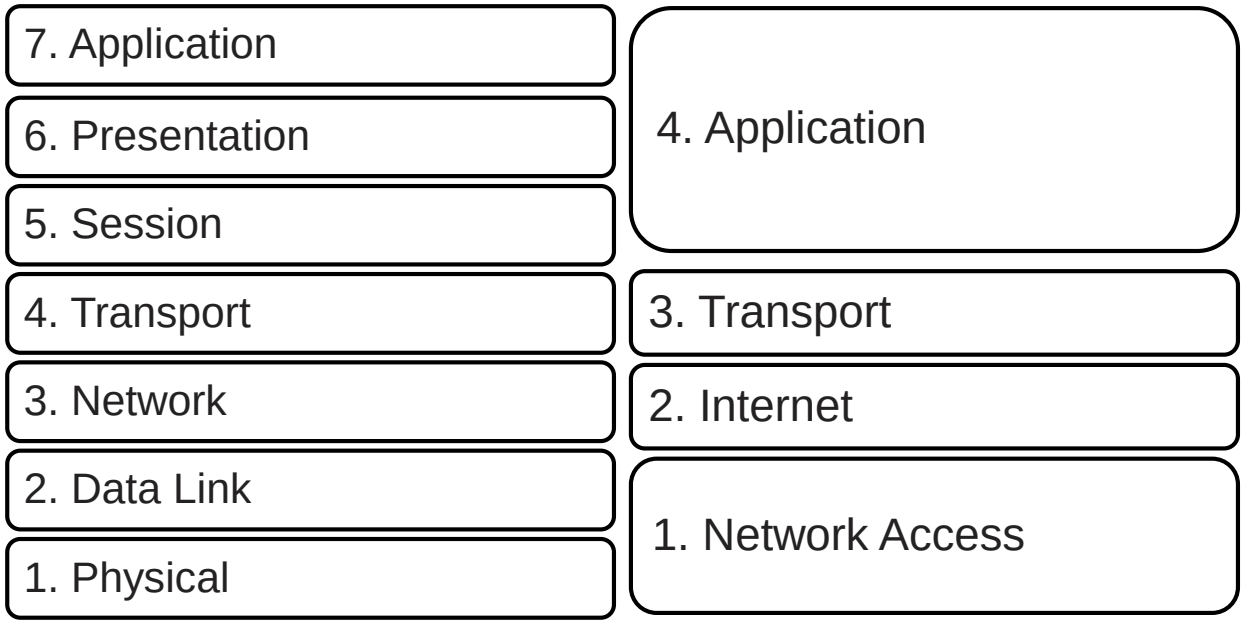
Why Bother with Packet Analysis?

- Faster than host-based forensics
- Investigate networking alerts
- Troubleshoot networking issues
- Tell if tools are working
- Reverse engineer protocols

The OSI Model



The TCP/IP Model



Encapsulation

Example encapsulated HTTP request

Decimal in a New Light

8080

$10^3 \mid 10^2 \mid 10^1 \mid 10^0$

$10^0 = 1$
$10^1 = 10$
$10^2 = 100$
$10^3 = 1000$
$10^4 = 10000$

Hexadecimal Numbers

- Base 16 (normal counting is decimal, base 10)
- Decimal to Hex : Divide by powers of 16, ignore remainders
- Hex to Decimal : Multiply by powers of 16
- Prefixed by 0x

Counting in Hex

Decimal	Hex
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

Decimal	Hex
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10

Decimal to Hex Conversion

8080/4096 = 1.xxxx
8080-(4096*1) = 3984
3984/256 = 15.xxxx
3984-(15*256) = 144
144/16 = 9.0

$16^0 = 1$
$16^1 = 16$
$16^2 = 256$
$16^3 = 4096$
$16^4 = 65536$

1 F 9 0

16^3| 16^2 | 16^1 | 16^0

Hex to Decimal Conversion

1 F 9 0

16^3| 16^2 | 16^1 | 16^0

1(4096) + 15(256) + 9(16) + 0(1) = ?

4096+3840+144+0 = 8080

$16^0 = 1$

$16^1 = 16$

$16^2 = 256$

$16^3 = 4096$

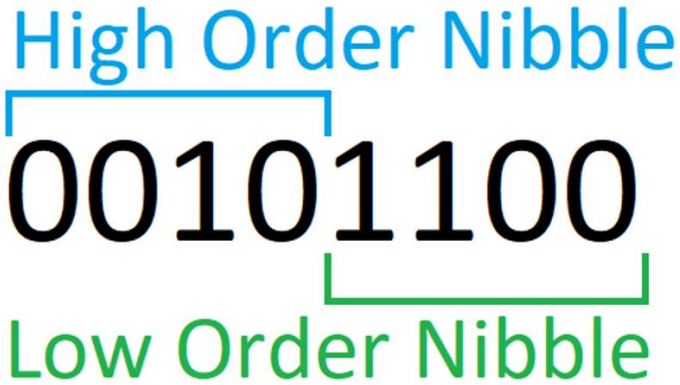
$16^4 = 65536$

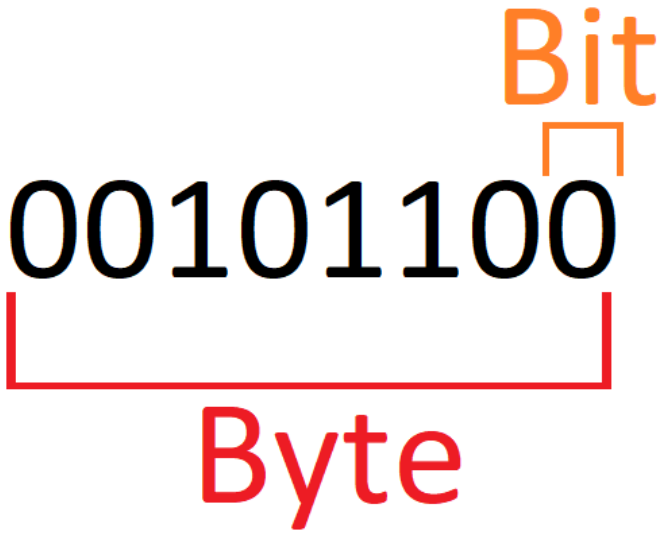
Binary Numbers

- Base 2
- Decimal to Binary : Divide by powers of 2, ignore remainders
- Binary to Decimal : Multiply by powers of 2
- Hex to Binary : Direct conversion of bytes to bits
- Binary to Hex : Direct conversion of bits to bytes

Bits, Bytes, and Nibbles

- A bit is a single binary number, 1 or 0
- A byte is 8 bits
- A nibble is 4 bits





Counting in Binary

Decimal	Hex	Binary
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000

Decimal	Hex	Binary
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	00010000

Hex to Binary

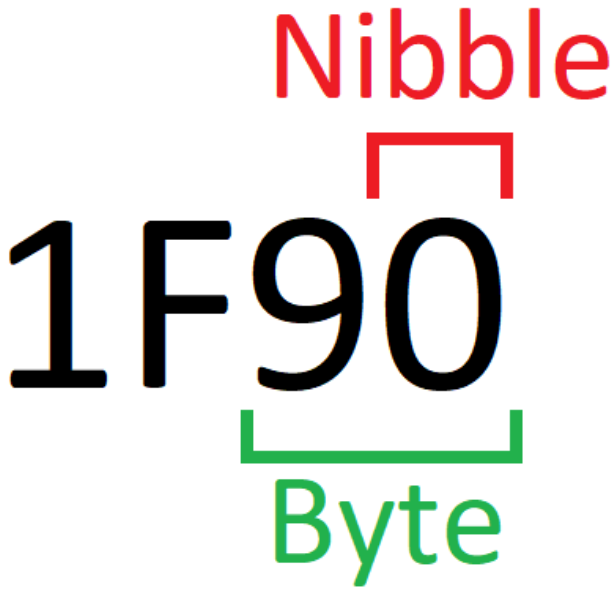
1F90=

0001 1111 1001 0000

1 = 0001
F = 1111
9 = 1001
0 = 0000

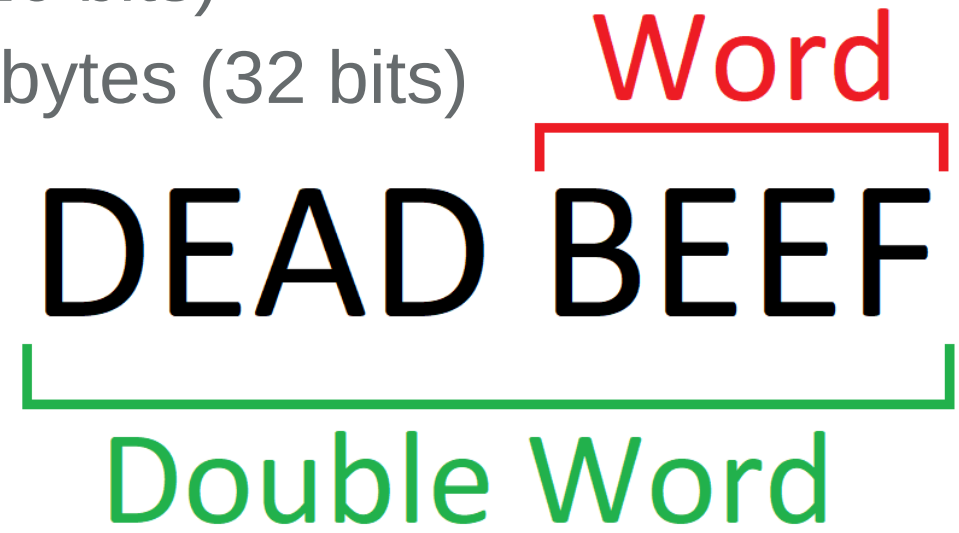
Hex Bytes and Nibbles

- Nibble is a single hex digit
- Byte is two hex digits



Words

- A word is 2 bytes (16 bits)
- A double-word is 4 bytes (32 bits)
- Often done in hex



Knowledge Check

- Convert CAFE to decimal
- Convert 7A69 to binary
  - What is the high order nibble of the low order byte in binary?
  - What is the low order nibble of the high order byte in hex?

Knowledge Check - Answers

- Convert CAFE to decimal – C=12, A=10, F=15, E=14

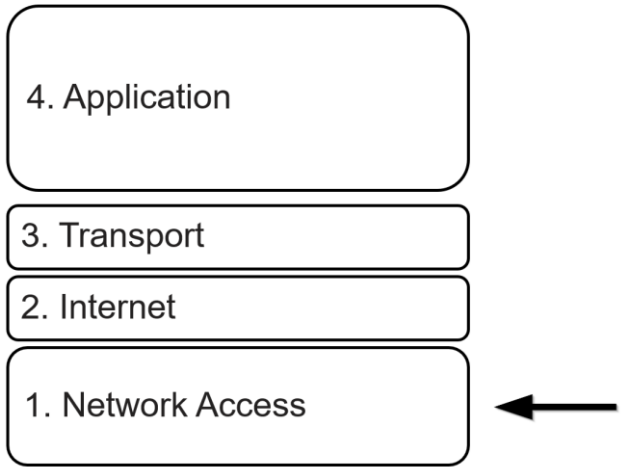
$12(16^3)+10(16^2)+15(16^1)+14(16^0)$ =

12*4096+10*256+15*16+14 = 51966

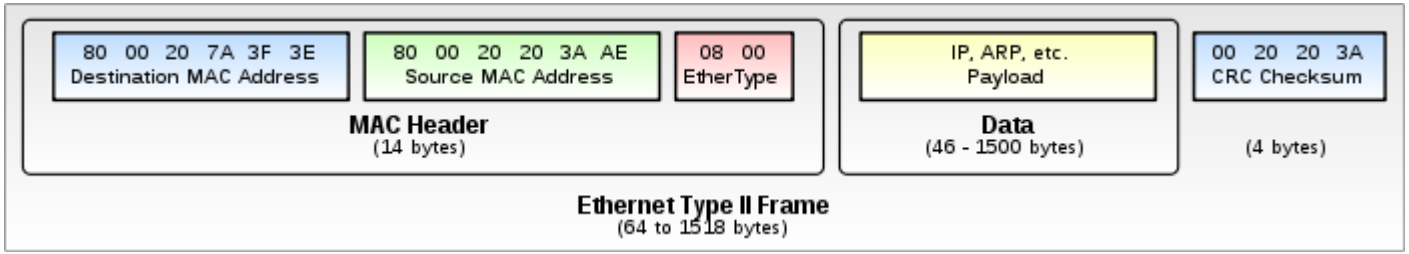
- Convert 7A69 to binary – 0111 1010 0110 1001
  - What is the high order nibble of the low order byte in binary?
    0110
  - What is the low order nibble of the high order byte in hex?
    A

Network Access Layer

- Common protocols: Ethernet, Wireless, Bluetooth
- Ethernet & Wireless use Media Access Control (MAC) addresses
- Information does not “route”



Ethernet Headers – Network Access Layer

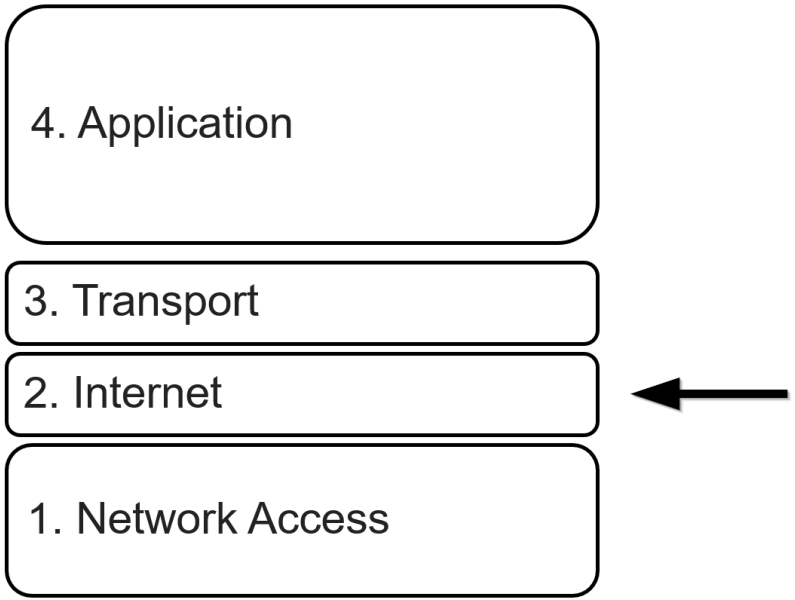


Source: https://en.wikipedia.org/wiki/Ethernet_frame

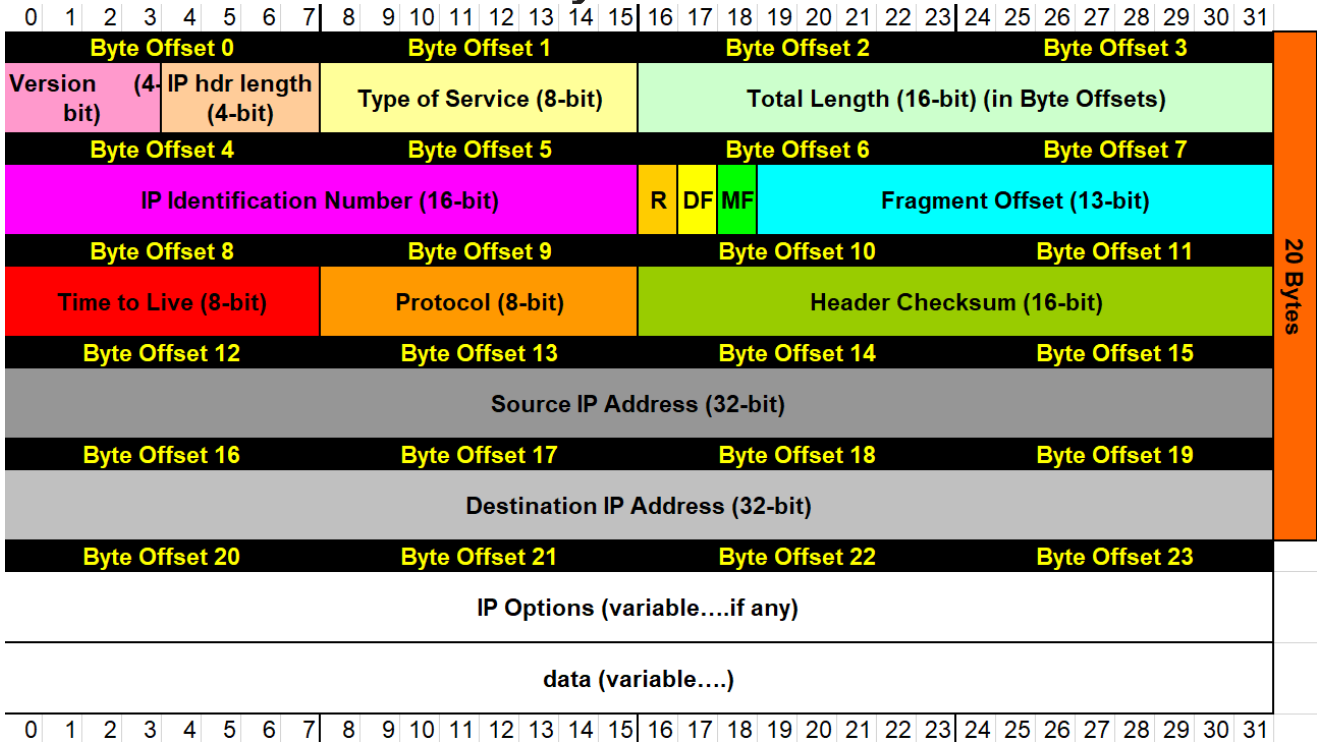
```
> Frame 56: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits)
v Ethernet II, Src: Vmware_f0:0b:61 (00:0c:29:f0:0b:61), Dst: AsustekC_be:f7:98 (08:60:6e:be:f7:98)
  > Destination: AsustekC_be:f7:98 (08:60:6e:be:f7:98)
  > Source: Vmware_f0:0b:61 (00:0c:29:f0:0b:61)
    Type: IPv4 (0x0800)

0000  08 60 6e be f7 98 00 0c  29 f0 0b 61 08 00 45 00   .`n..... )..a..E.
0010  00 a1 6e 5e 40 00 40 06  41 4b c0 a8 02 7a 36 98   ..n^@.@. AK...z6.
0020  90 f3 92 4c 00 50 71 0d  f2 d6 52 61 85 9d 80 18   ...L.Pq. ..Ra....
0030  00 e5 8b 41 00 00 01 01  08 0a 01 5a 9c 93 44 5f   ...A.... ...Z..D_
0040  16 d7 47 45 54 20 2f 66  34 39 30 61 33 35 61 63   ..GET /f 490a35ac
```

Internet Layer – IPv4

- Routes packets across the internet
- Uses 32-bit IP addresses
- ARP resolves MAC to IP
- Header length depends on options
  - Not that common
- Does not guarantee delivery



IPv4 Header – Internet Layer



http://wiki.gnhlug.org/twiki2/pub/Www/IpReference/Packet_Headers_Subnet_Breakdown.xls

IPv4 Fragmentation

- Naturally occurs
  - Packet size exceeds MTU (Maximum Transmission Unit)
- Sometimes orgs block it
- Foundation for many bypass techniques
- Observation point and target must decode the same way
- RFCs do not deal with edge cases

IPv4 Fragmentation

- Fragments are in IP header, bytes 6 and 7
- DF – Don’t Fragment
- MF – More Fragments

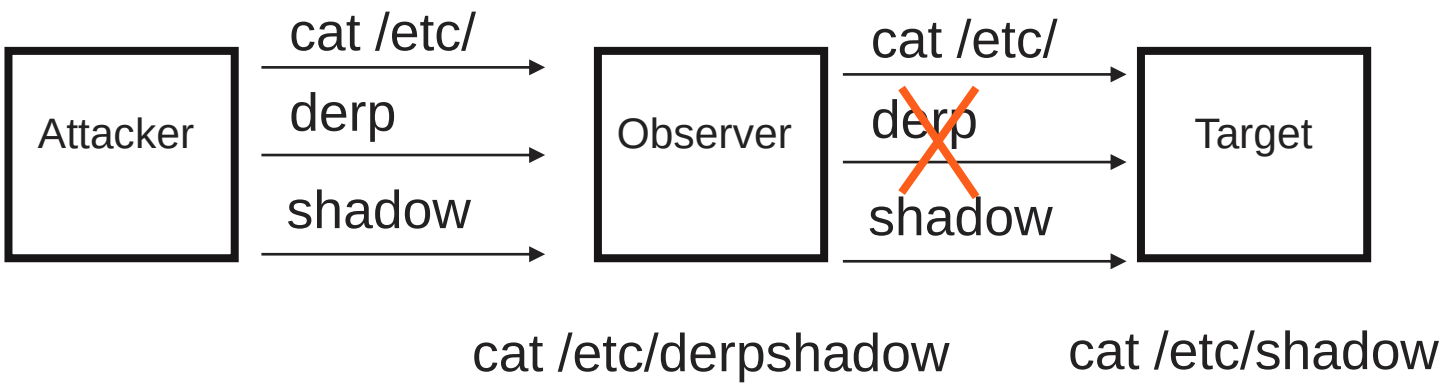
- Multiply fragment offset by 8, 13 bits cannot represent full packet length
- MF bit set when more fragments follow (not in last frag or unfragmented packets)

IPv4 Fragmentation – Example (icmpfrag.pcap)

- Payload per packet size = MTU-(IP header length)
- Only first packet in the train has the transport protocol header
- MTU of 1500 (Ethernet)
- 20 byte IP header
- 1480 payload bytes per packet
- Packet 1 Offset : 0
- Packet 2 Offset : 1480/8 decimal = B9 hex
- Packet 3 Offset : 2960/8 decimal = 172 hex

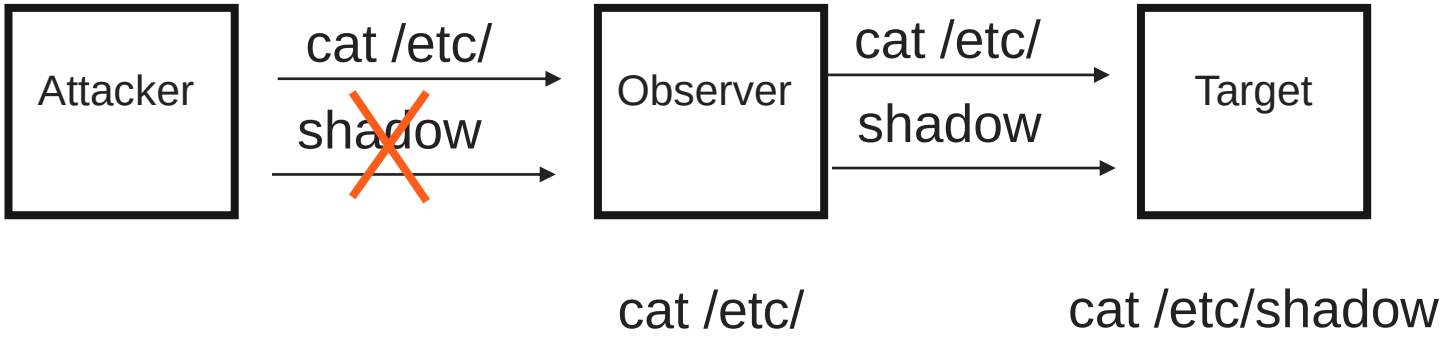
IPv4 Fragmentation - Insertion

- Observation system accepts data the target does not



IPv4 Fragmentation - Evasion

- Observation system rejects data the target does not

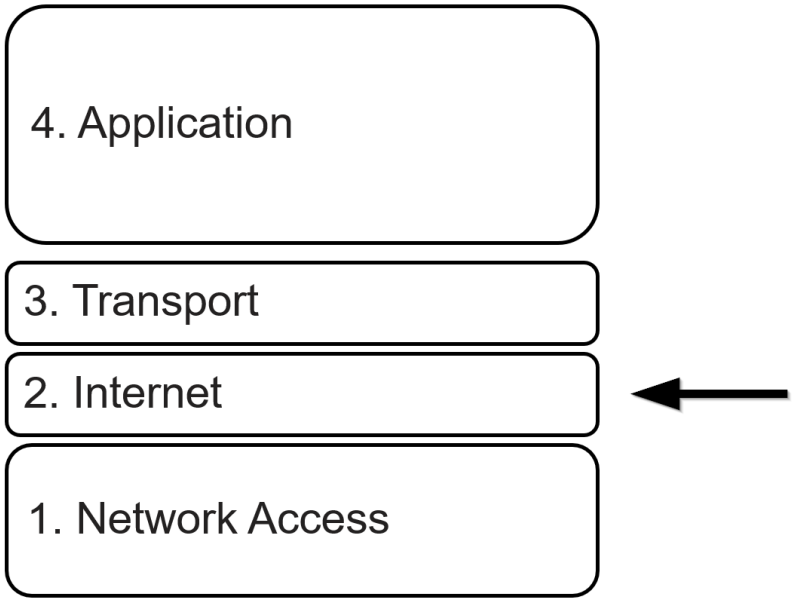


Traceroute

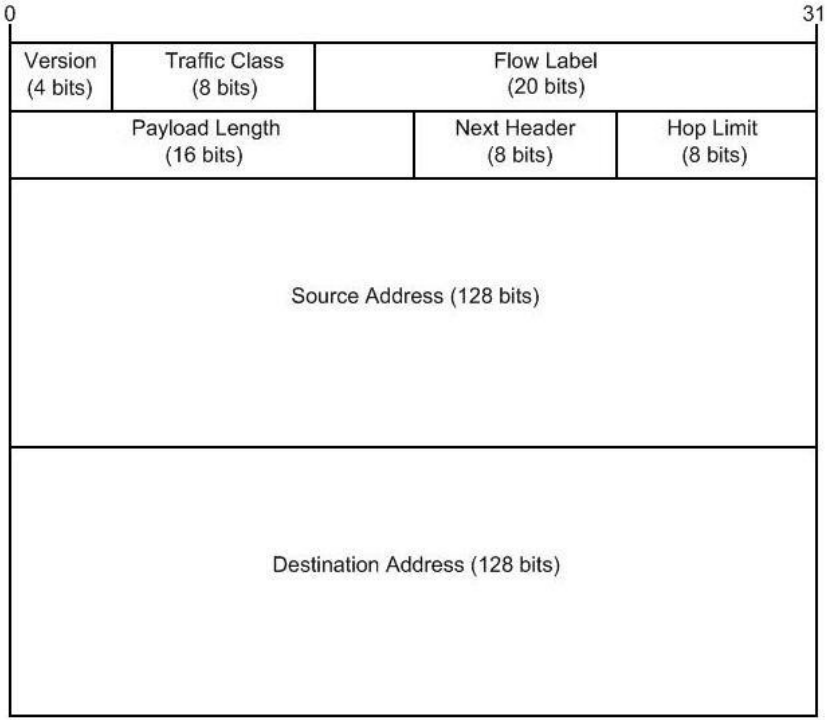
- Find list of hops between hosts
- Relies on ICMP Time Exceeded in Transit messages
- Can use ANY layer 4 protocol
- Send packets with variable TTL values
- Each hop decrements TTL by 1, discarding router sends ICMP
- Windows uses ICMP echo requests
- Linux uses UDP 33434 to UDP 33534

Internet Layer – IPv6

- Same purpose as IPv4
- Uses 128-bit IP addresses
- Neighbor Solicitation resolves MAC to IP
- Does not guarantee delivery



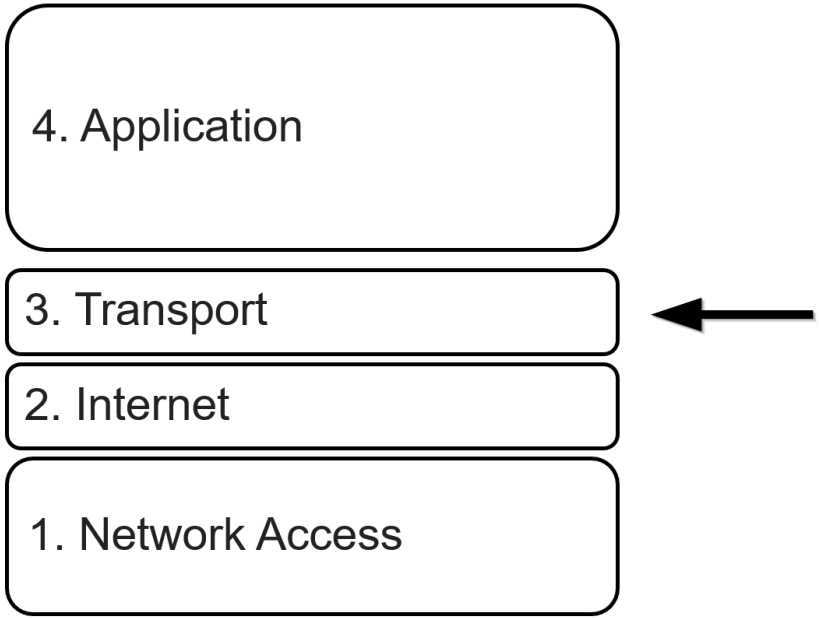
IPv6 Header – Internet Layer



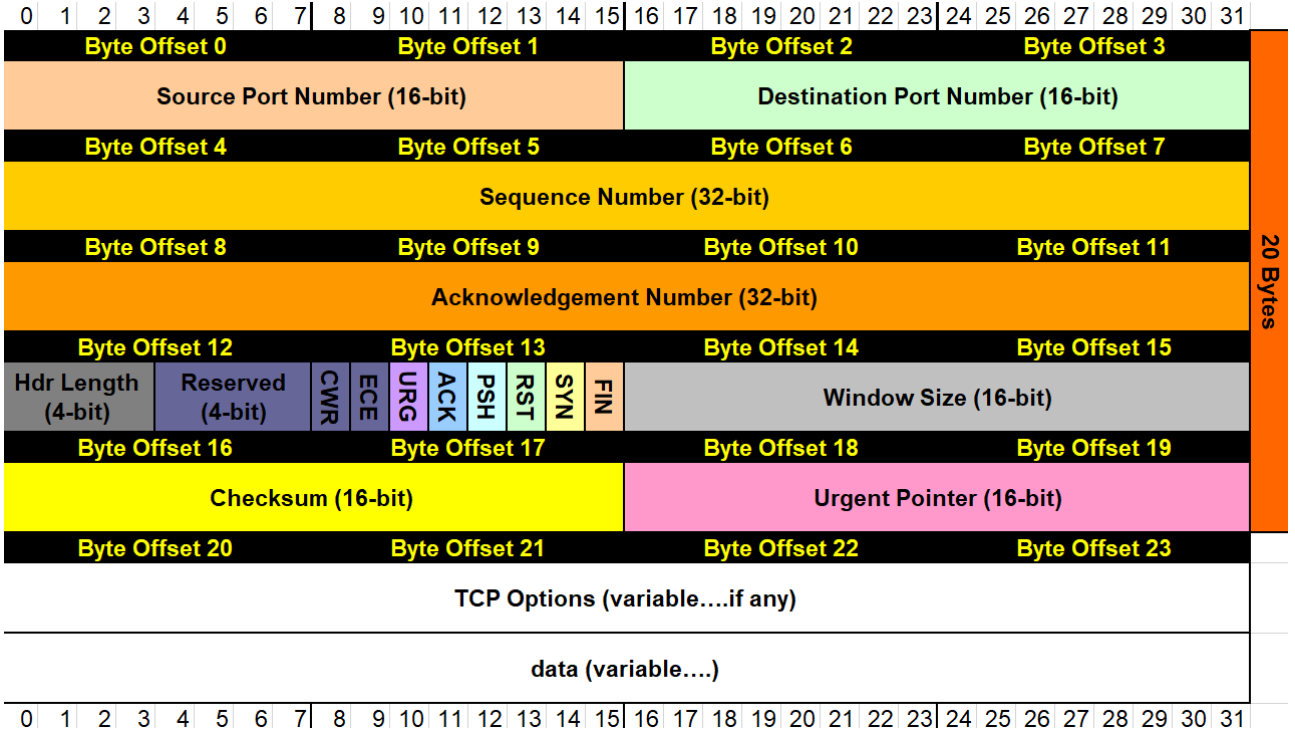
https://www.cisco.com/c/en/us/about/security-center/countermeasures-ipv6-type-0.html

Transport Layer - TCP

- Transmission Control Protocol
- Ensures packet delivery
- Most common transport layer protocol
- Communication status controlled with TCP flags

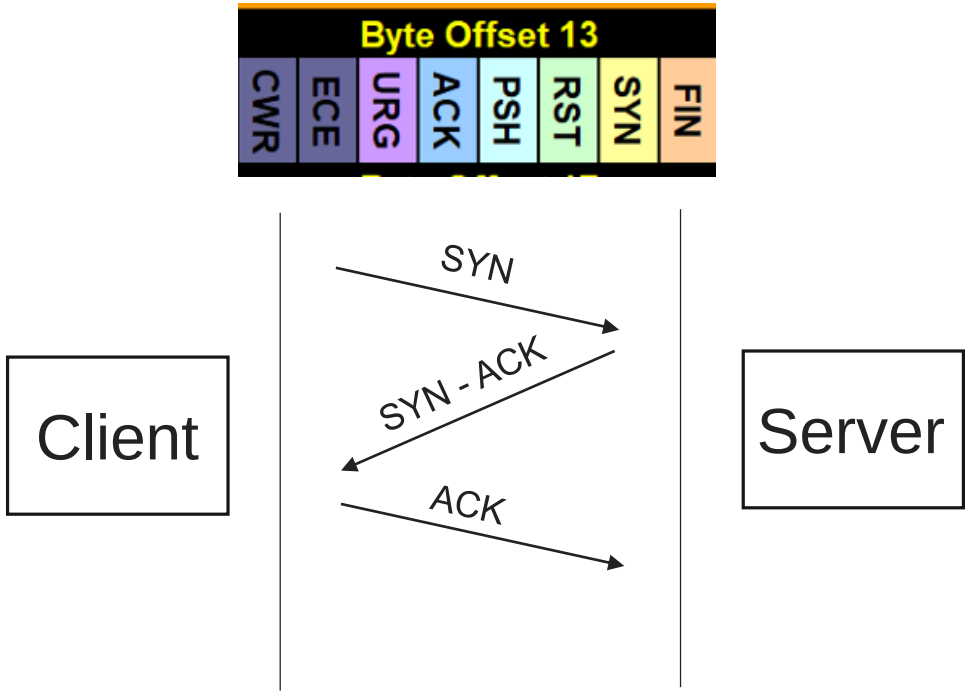


TCP Header

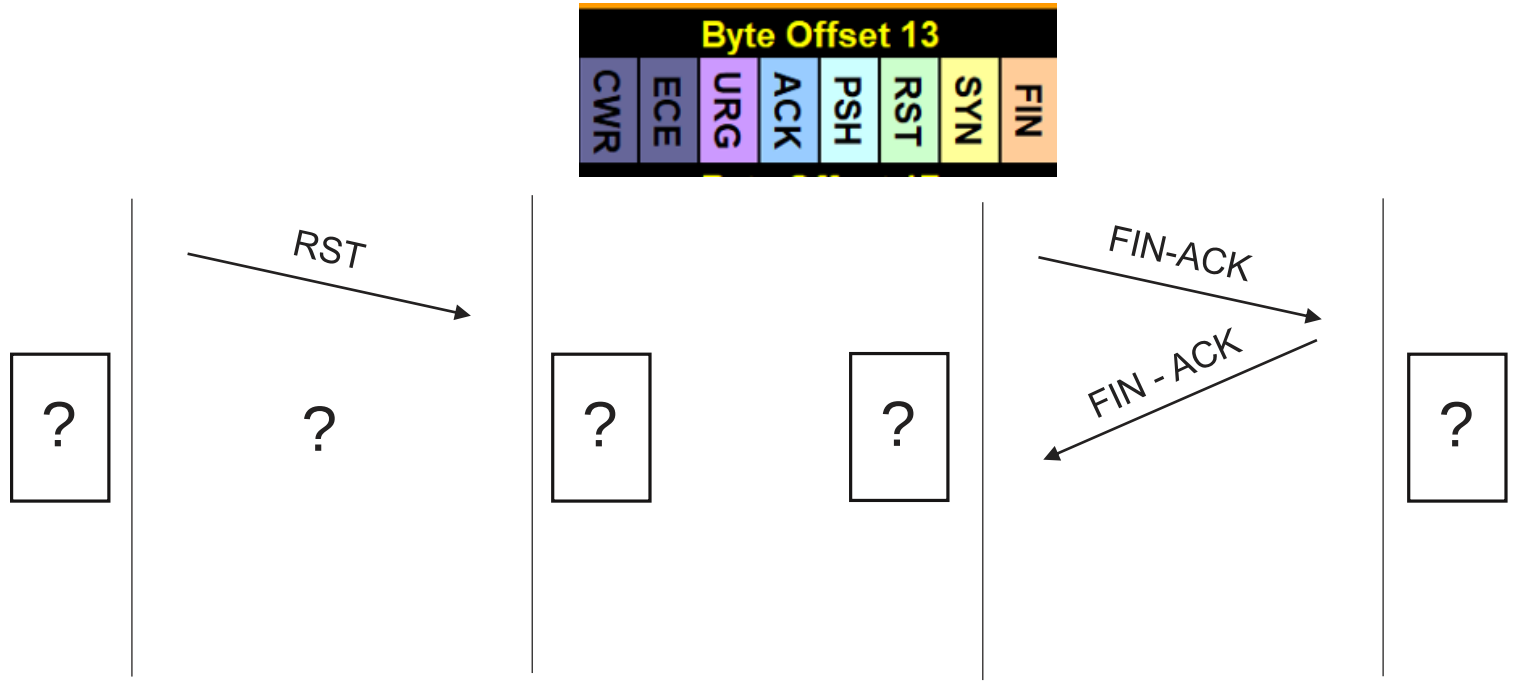


http://wiki.gnhlug.org/twiki2/pub/Www/IpReference/Packet_Headers_Subnet_Breakdown.xls

TCP Conversations - Beginning



TCP Conversations - Ending



TCP - Tracking Conversations

- Socket (Source – Destination combination)
- Sequence Number
  - Client and server have different sequence numbers
- Acknowledgement Number
  - Acknowledge last sequence # + payload length
- Wireshark tracks this well

TCP - Tracking Conversations

- C-S – Syn – SEQ 0x82acd14d ACK 0x00000000, no payload
- S-C – Syn-Ack – SEQ 0x9696fd7e ACK 0x82acd14e, no payload
- C-S – Ack – SEQ 0x82acd14e ACK 0x9696fd7f, no payload
- C-S – Ack – SEQ 0x82acd14e ACK 0x9696fd7f, 306 byte payload
- S-C – Ack – SEQ 0x9696fd7f ACK 0x82acd280, no payload
- S-C – Ack – SEQ 0x9696fd7f ACK 0x82acd280, 1261 byte payload
- C-S – Ack – SEQ 0x82acd280 ACK 0x9697026c, 286 byte payload
- S-C – Ack – SEQ 0x9697026c ACK 0x82acd39e, 1460 byte payload

TCP - Retransmission

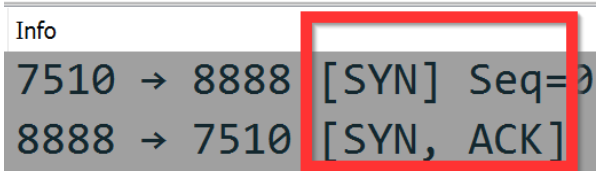
- Blocked connections usually reset or dropped
- Dropped packets look “lost” to host
- Hosts retry at progressively larger intervals

TCP - Retransmission

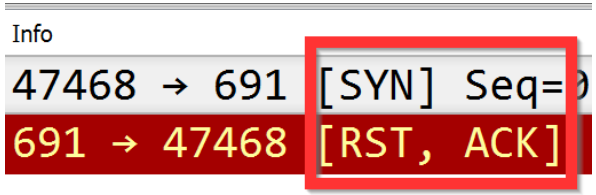
```
20:51:22.704756 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567562150 ecr 0,nop,wscale 10], length 0
20:51:23.725627 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567563176 ecr 0,nop,wscale 10], length 0
20:51:25.741171 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567565201 ecr 0,nop,wscale 10], length 0
20:51:29.869389 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567569352 ecr 0,nop,wscale 10], length 0
20:51:38.061189 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567577594 ecr 0,nop,wscale 10], length 0
20:51:54.189403 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567593827 ecr 0,nop,wscale 10], length 0
20:52:27.879042 IP 192.168.2.175.53840 > 172.217.0.4.6667: Flags [S], seq 2001742320,
[mss 1460,sackOK,TS val 2567627335 ecr 0,nop,wscale 10], length 0
```

TCP – “Normal” Behavior

- SYN to an open port causes SYN-ACK response

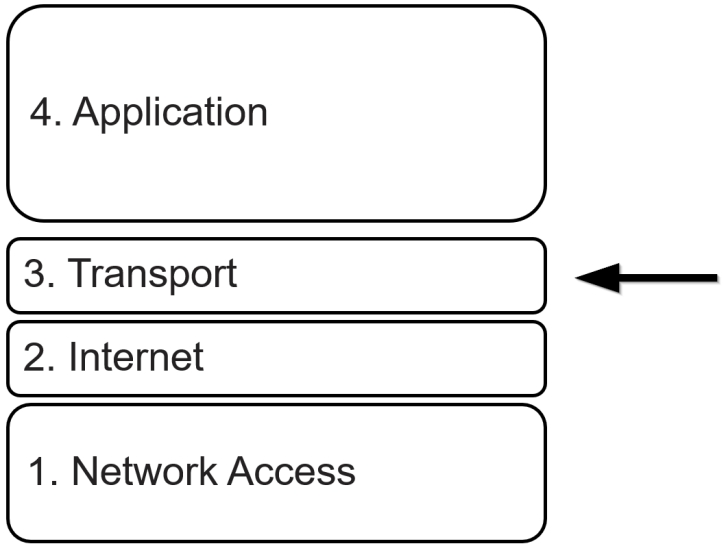


- SYN to a closed port causes a RST

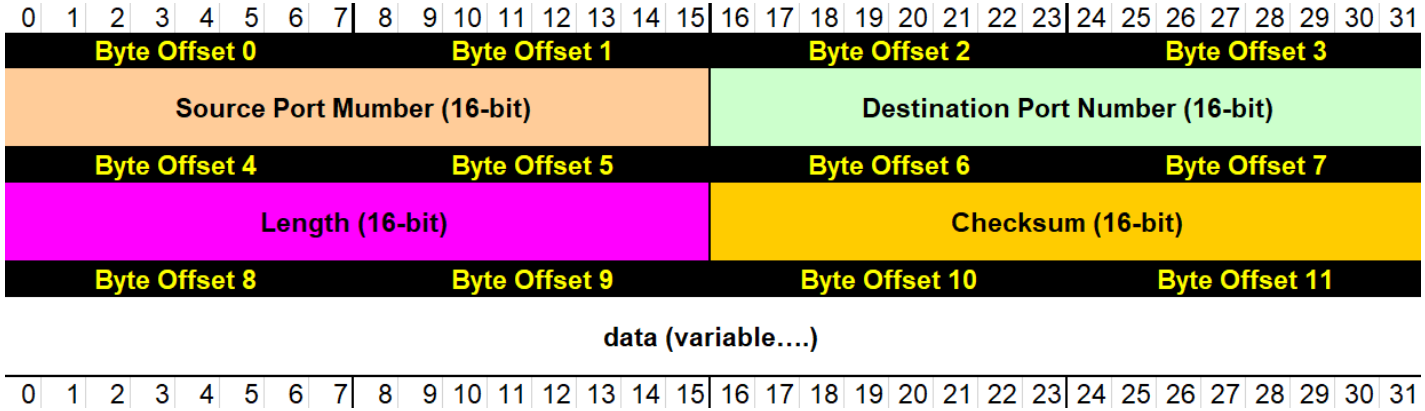


Transport Layer - UDP

- User Datagram Protocol
- Unreliable delivery
- Faster than TCP
- Packet loss handled at the application layer
- Suitable for streaming (VOIP, video, etc)
- DNS and tftp commonly use UDP
- QUIC uses UDP



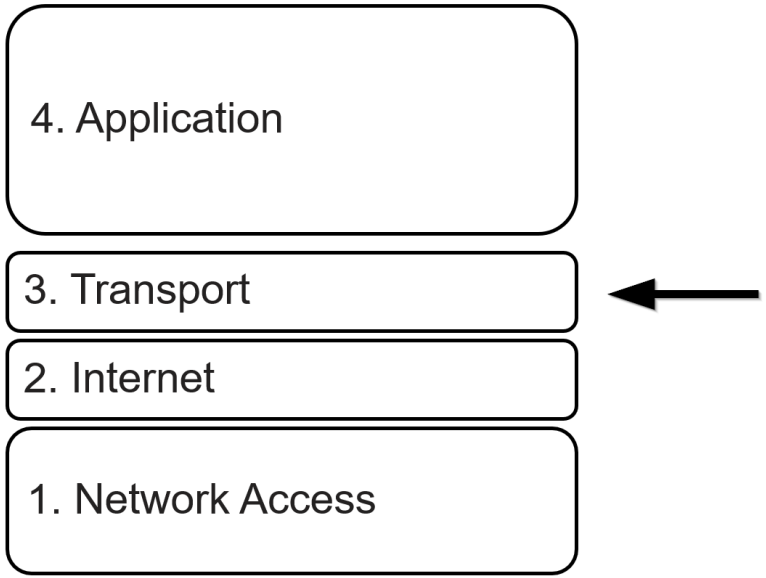
UDP Header



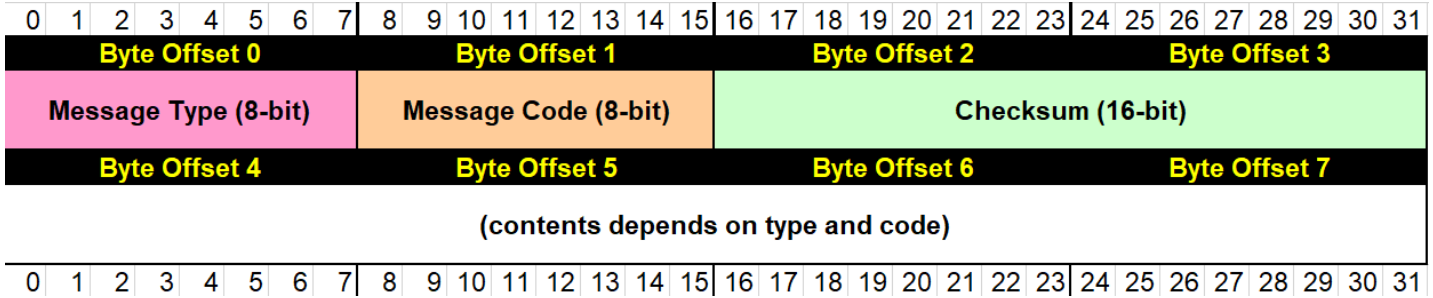
http://wiki.gnhlug.org/twiki2/pub/Www/IpReference/Packet_Headers_Subnet_Breakdown.xls

Transport Layer - ICMP

- Internet Control Message Protocol
- Administrative communication
- Ping, unreachable messages, router advertisement, redirection, etc.
- Often result from a stimulus
  - Ping response
  - Port unreachable



ICMP Header



http://wiki.gnhlug.org/twiki2/pub/Www/IpReference/Packet_Headers_Subnet_Breakdown.xls

ICMP Header

TYPE	CODE	Description
0	0	Echo Reply
3	0	Network Unreachable
3	1	Host Unreachable
3	2	Protocol Unreachable
3	3	Port Unreachable
3	4	Fragmentation needed but no frag. bit set
3	5	Source routing failed
3	6	Destination network unknown
3	7	Destination host unknown
3	8	Source host isolated (obsolete)
3	9	Destination network administratively prohibited
3	10	Destination host administratively prohibited
3	11	Network unreachable for TOS
3	12	Host unreachable for TOS
3	13	Communication administratively prohibited by filtering
3	14	Host precedence violation
3	15	Precedence cutoff in effect

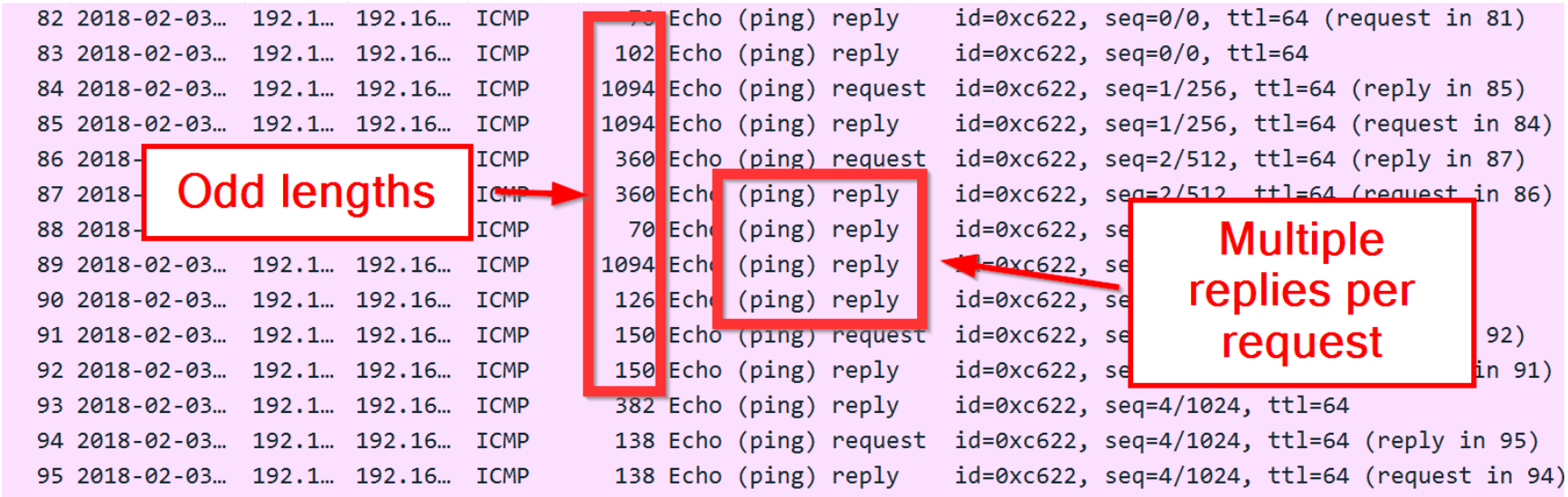
TYPE	CODE	Description
4	0	Source quench
5	0	Redirect for network
5	1	Redirect for host
5	2	Redirect for TOS and network
5	3	Redirect for TOS and host
8	0	Echo request
9	0	Router advertisement
10	0	Route solicitation
11	0	TTL equals 0 during transit
11	1	TTL equals 0 during reassembly
12	0	IP header bad (catchall error)
12	1	Required options missing
13	0	Timestamp request (obsolete)
14		Timestamp reply (obsolete)
15	0	Information request (obsolete)
16	0	Information reply (obsolete)
17	0	Address mask request
18	0	Address mask reply

http://slideplayer.com/slide/6252793/

ICMP Anomalous Traffic

- Response with no stimulus
  - ICMP echo responses with no request
  - Different numbers of ICMP echo requests and responses
  - Nonstandard payloads
    - Windows echo request abcdefghijklmnopqrstuvwabcdefghi
    - Linux echo request !"#$%&'()*+,-./01234567

ICMP Tunnel


82 2018-02-03… 192.1… 192.16… ICMP 70 Echo (ping) reply id=0xc622, seq=0/0, ttl=64 (request in 81)
83 2018-02-03… 192.1… 192.16… ICMP 102 Echo (ping) reply id=0xc622, seq=0/0, ttl=64
84 2018-02-03… 192.1… 192.16… ICMP 1094 Echo (ping) request id=0xc622, seq=1/256, ttl=64 (reply in 85)
85 2018-02-03… 192.1… 192.16… ICMP 1094 Echo (ping) reply id=0xc622, seq=1/256, ttl=64 (request in 84)
86 2018- ICMP 360 Echo (ping) request id=0xc622, seq=2/512, ttl=64 (reply in 87)
87 2018- ICMP 360 Echo (ping) reply id=0xc622, seq=2/512, ttl=64 (request in 86)
88 2018- ICMP 70 Echo (ping) reply id=0xc622, se
89 2018-02-03… 192.1… 192.16… ICMP 1094 Echo (ping) reply id=0xc622, se
90 2018-02-03… 192.1… 192.16… ICMP 126 Echo (ping) reply id=0xc622, se
91 2018-02-03… 192.1… 192.16… ICMP 150 Echo (ping) request id=0xc622, se 92)
92 2018-02-03… 192.1… 192.16… ICMP 150 Echo (ping) reply id=0xc622, se in 91)
93 2018-02-03… 192.1… 192.16… ICMP 382 Echo (ping) reply id=0xc622, seq=4/1024, ttl=64
94 2018-02-03… 192.1… 192.16… ICMP 138 Echo (ping) request id=0xc622, seq=4/1024, ttl=64 (reply in 95)
95 2018-02-03… 192.1… 192.16… ICMP 138 Echo (ping) reply id=0xc622, seq=4/1024, ttl=64 (request in 94)
Odd lengths
Multiple replies per request

Transport Layer – ICMPv6

- Internet Control Message Protocol
- Same purpose as ICMPv4
- Additional responsibility – Neighbor Discovery Protocol

ICMPv6 packet

Bit offset	0–7	8–15	16–31
0	Type	Code	Checksum
32	Message body		

Knowledge Check

- What ICMP type and code indicate a ping response?
- What two bytes start most IPv4 packets?
- What ICMP type might indicate a traceroute?
- What are the steps to the TCP 3-way handshake?
- What are the two common methods of terminating a TCP connection?

Knowledge Check

- What ICMP type and code indicate a ping response?
  - Type 0, code 0
- What two bytes start most IPv4 packets?
  - 0x4500
- What ICMP type might indicate a traceroute?
  - Type 11
- What are the sets of flags in a normal TCP 3-way handshake?
  - SYN, SYN-ACK, ACK
- What are the two common methods of terminating a TCP connection?
  - FIN, RST

Part 2 – Tools

Overview
Part 2 – Tools

- TCPDump
- Wireshark
- TShark
- ngrep
- GNU Tools

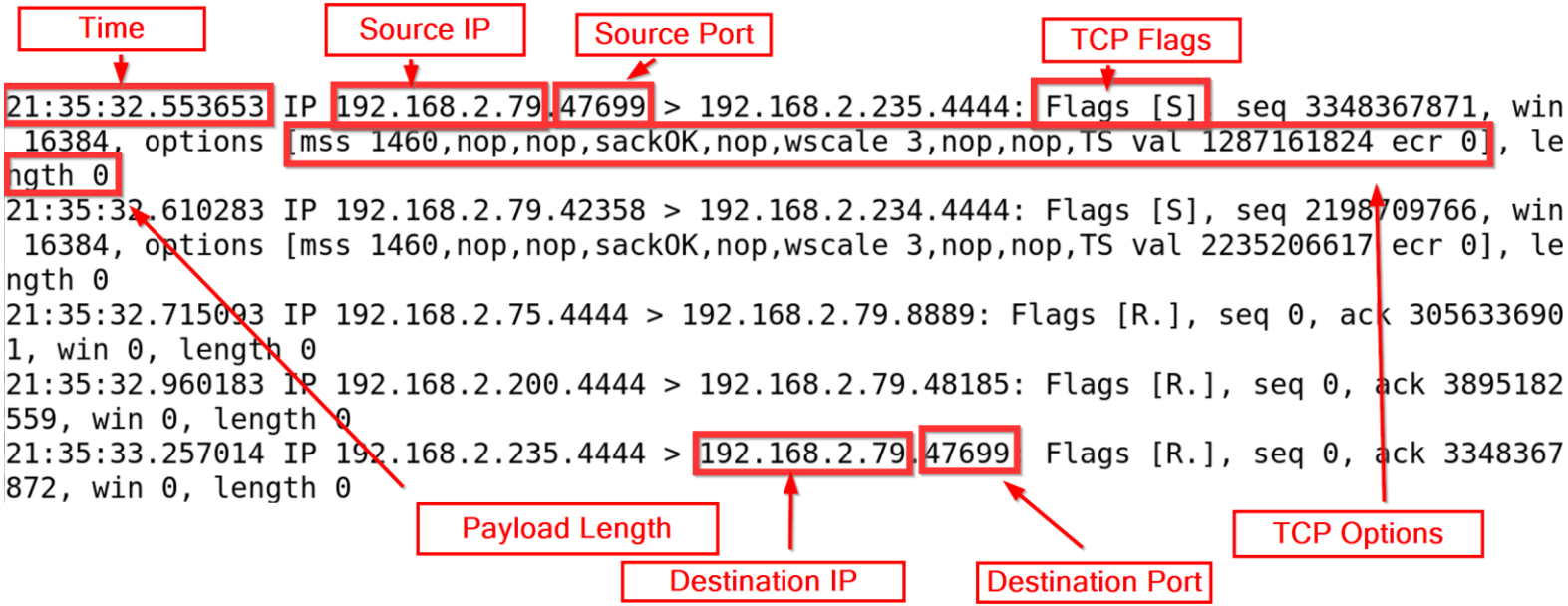
TCPDump

- Packet sniffing application
- *nix, windump on Windows
- Little interpretation past transport layer
- Fastest sniffer available
- Uses Berkeley Packet Filters (BPF)

TCPDump Common Options

-nn : Do not resolve host names or ports
-i : Capture interface
-w : write packets to a file
-r : read capture file
-A : Dump ASCII
-x : Dump hex
-X : Dump hex and ASCII
-v[vvvv] : increase verbosity

TCPDump


Time
Source IP
Source Port
TCP Flags
21:35:32.553653 IP 192.168.2.79.47699 > 192.168.2.235.4444: Flags [S], seq 3348367871, win
16384, options [mss 1460,nop,nop,sackOK,nop,wscale 3,nop,nop,TS val 1287161824 ecr 0], le
ngth 0
21:35:32.610283 IP 192.168.2.79.42358 > 192.168.2.234.4444: Flags [S], seq 2198709766, win
16384, options [mss 1460,nop,nop,sackOK,nop,wscale 3,nop,nop,TS val 2235206617 ecr 0], le
ngth 0
21:35:32.715093 IP 192.168.2.75.4444 > 192.168.2.79.8889: Flags [R.], seq 0, ack 305633690
1, win 0, length 0
21:35:32.960183 IP 192.168.2.200.4444 > 192.168.2.79.48185: Flags [R.], seq 0, ack 3895182
559, win 0, length 0
21:35:33.257014 IP 192.168.2.235.4444 > 192.168.2.79.47699: Flags [R.], seq 0, ack 3348367
872, win 0, length 0
Payload Length
Destination IP
Destination Port
TCP Options

TCPDump Verbose Output

```
21:51:04.312150 IP (tos 0x0, ttl 47, id 52284, offset 0, flags [DF], proto UDP (17), lengt
h 76)
    173.71.68.101.123 > 192.168.2.175.51305: [udp sum ok] NTPv4, length 48
        Server, Leap indicator:  (0), Stratum 1 (primary reference), poll 3 (8s), precisio
n -19
        Root Delay: 0.000000, Root dispersion: 0.050003, Reference-ID: PPS^@
          Reference Timestamp:  3726528663.424345441 (2018/02/01 21:51:03)
          Originator Timestamp: 3726528664.061811511 (2018/02/01 21:51:04)
          Receive Timestamp:    3726528664.819150386 (2018/02/01 21:51:04)
          Transmit Timestamp:   3726528664.819465378 (2018/02/01 21:51:04)
            Originator - Receive Timestamp:  +0.757338874
            Originator - Transmit Timestamp: +0.757653867
```

Berkeley Packet Filters

- Widely used in networking tools
- Limit traffic TCPDump accepts
- Allows targeted captures
- Filter existing captures

Berkeley Packet Filters

- host *host* : capture traffic from a specific host
- src host *host* : specific source host
- dst host *host* : specific destination host
- ip src host *host* : IP packets from specific source host
- Port *port* : Packets to or from a specific transport-layer port
- ether *ehost* : packets from a specific MAC
- net *subnet* : packets to or form a specific subnet

Berkeley Packet Filters

- ip[0] = 0x45 : IPv4 packets, header length 20
- tcp[2:2] = 443 : TCP packets destination port 443
- tcp : capture all TCP packets
- udp : capture all UDP packets
- tcp && src host *host* : TCP packets from a specific host
- tcp || src host *host* : Any TCP packet or packet from a specific host

TCP Flags (Byte 13)

- Byte 13 in the TCP header contains control flags
- Help manage the TCP conversation

SYN Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	0	0	0	1	0

Berkeley Packet Filters

- tcp[13] = 0x02 : SYN packets
- tcp[13] = 0x04 : RST packets
- tcp[tcpflags] = tcp-syn : also SYN packets

- How can you capture packets without the SYN flag set?

Bit Masking

- Logical AND (&)
  - Both statements must be true
- Logical OR (|)
  - Either statement is true

	False	True
False	False	False
True	False	True

	False	True
False	False	True
True	True	True

Berkeley Packet Filters

- Packets **with** the SYN flag set
- tcp[13] & 0x02 == 0x02

SYN Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0

SYN-ACK Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0

SYN-FIN Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0

XMAS Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	0

Berkeley Packet Filters

- Packets **without** the SYN flag set
- tcp[13] & 0x02 != 0x02
- tcp[13] & 0x02 = 0

RST Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0

FIN-ACK Packet Flags							
CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0

Knowledge Check

- Find SYN-ACK packets
- Find packets with a source port of 80
- Find packets with all TCP flags set
- Find all UDP packets with a destination port of 123
- Find all type 3 code 3 ICMP packets

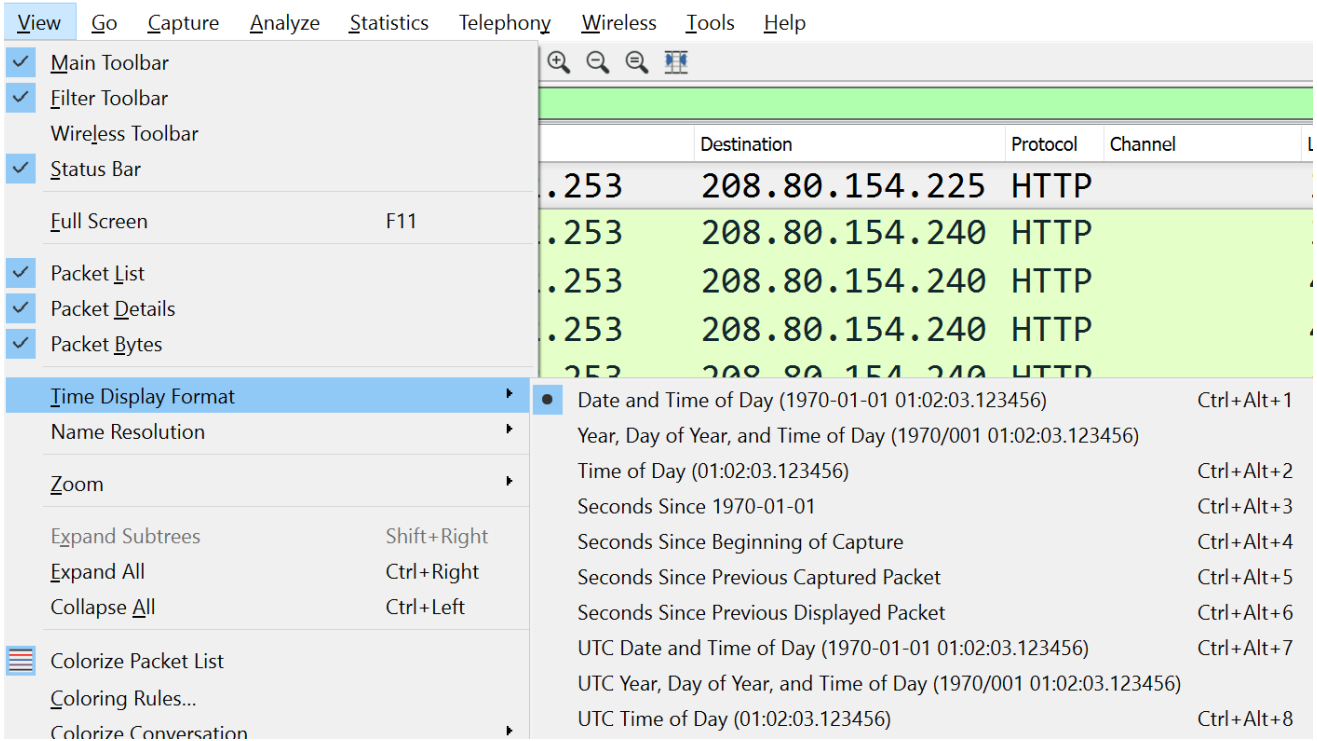
Knowledge Check Answers

- Find SYN-ACK packets
  - tcp[13]=0x12
- Find packets with a source port of 80
  - src port 80
  - udp[0:2]=80 || tcp[0:2]=80
- Find packets with all TCP flags set
  - tcp[13]=0xff
- Find all UDP packets with a destination port of 123
  - udp[2:2]=123
  - udp && dst port 123
- Find all type 3 code 3 ICMP packets
  - icmp[0]=3 && icmp[1]=3

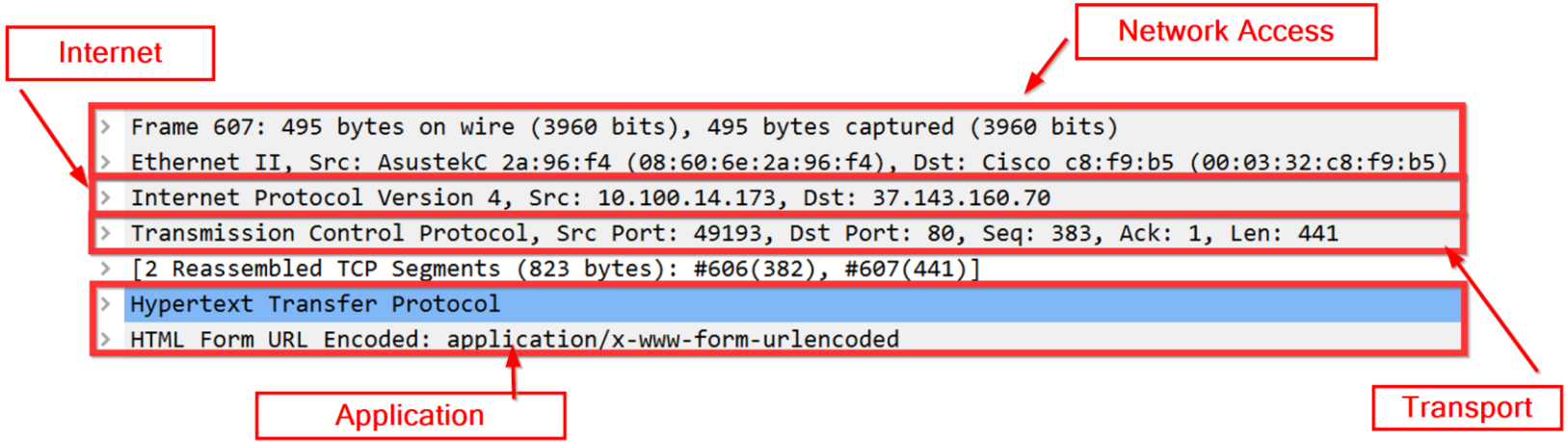
Wireshark

- Not as confusing as TCPDump
- Graphical interface to view packets
- Robust application layer support
- More powerful filters
- Slow.
- Can be misleading
- Almost unusable > 1GB packet captures
- Can have remote code execution vulnerabilities

Wireshark – Time Display



Wireshark


Internet
Network Access
Frame 607: 495 bytes on wire (3960 bits), 495 bytes captured (3960 bits)
Ethernet II, Src: AsustekC 2a:96:f4 (08:60:6e:2a:96:f4), Dst: Cisco c8:f9:b5 (00:03:32:c8:f9:b5)
Internet Protocol Version 4, Src: 10.100.14.173, Dst: 37.143.160.70
Transmission Control Protocol, Src Port: 49193, Dst Port: 80, Seq: 383, Ack: 1, Len: 441
[2 Reassembled TCP Segments (823 bytes): #606(382), #607(441)]
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
Application
Transport

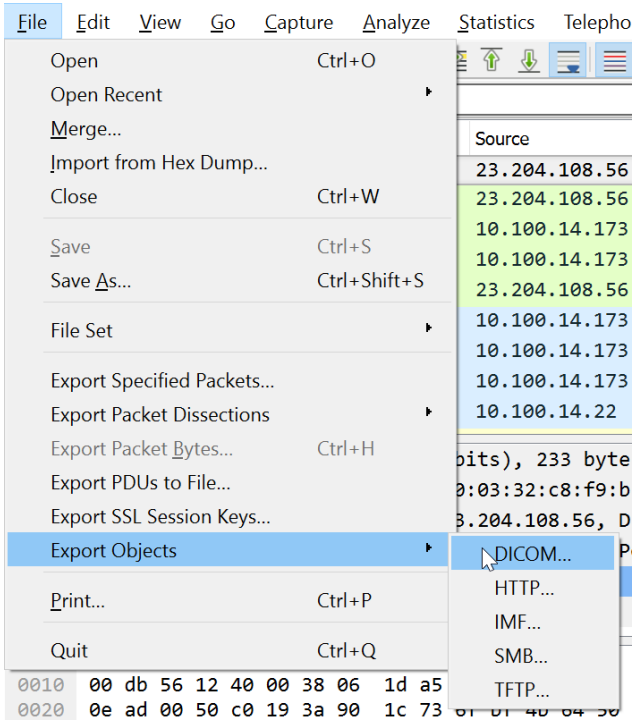
Wireshark Statistics

Ethernet · 5 | IPv4 · 20 | IPv6 | TCP · 65 | UDP · 46

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Latitude	Longitude
37.143.160.70	80	12	1668	5	431	7	1237	—	—
47.89.250.83	80	12	3325	5	2070	7	1255	—	—
83.255.177.54	80	12	2302	5	1065	7	1237	—	—
5.56.73.146	80	11	2536	5	1359	6	1177	—	—
41.110.200.194	80	11	2402	5	1225	6	1177	—	—
46.29.1.191	80	11	2339	5	1162	6	1177	—	—
81.12.175.59	80	11	1608	5	431	6	1177	—	—
84.224.160.159	80	11	1608	5	431	6	1177	—	—
109.96.148.33	80	11	1608	5	431	6	1177	—	—
195.228.41.2	80	11	1608	5	431	6	1177	—	—
23.204.108.56	80	10	858	5	461	5	397	—	—
2.185.239.164	80	3	194	0	0	3	194	—	—
10.100.14.22	88	201	62 k	94	30 k	107	32 k	—	—
10.100.14.22	135	30	3720	12	1672	18	2048	—	—
10.100.14.22	389	79	25 k	35	11 k	44	14 k	—	—
10.100.14.22	445	104	24 k	49	8012	55	16 k	—	—

Wireshark Exports

- Dump files from pcaps
- File | Export Objects
- Limited protocol support
  - HTTP
  - SMB
  - DICOM
  - IMF
  - TFTP

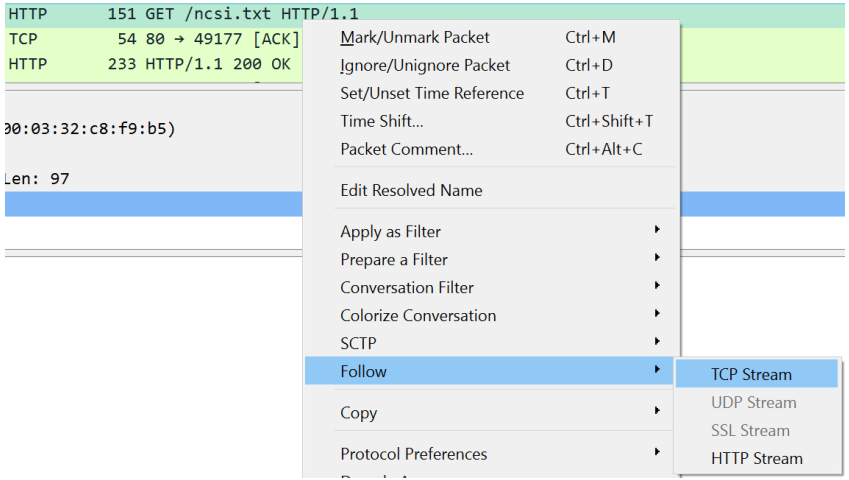


Export Objects

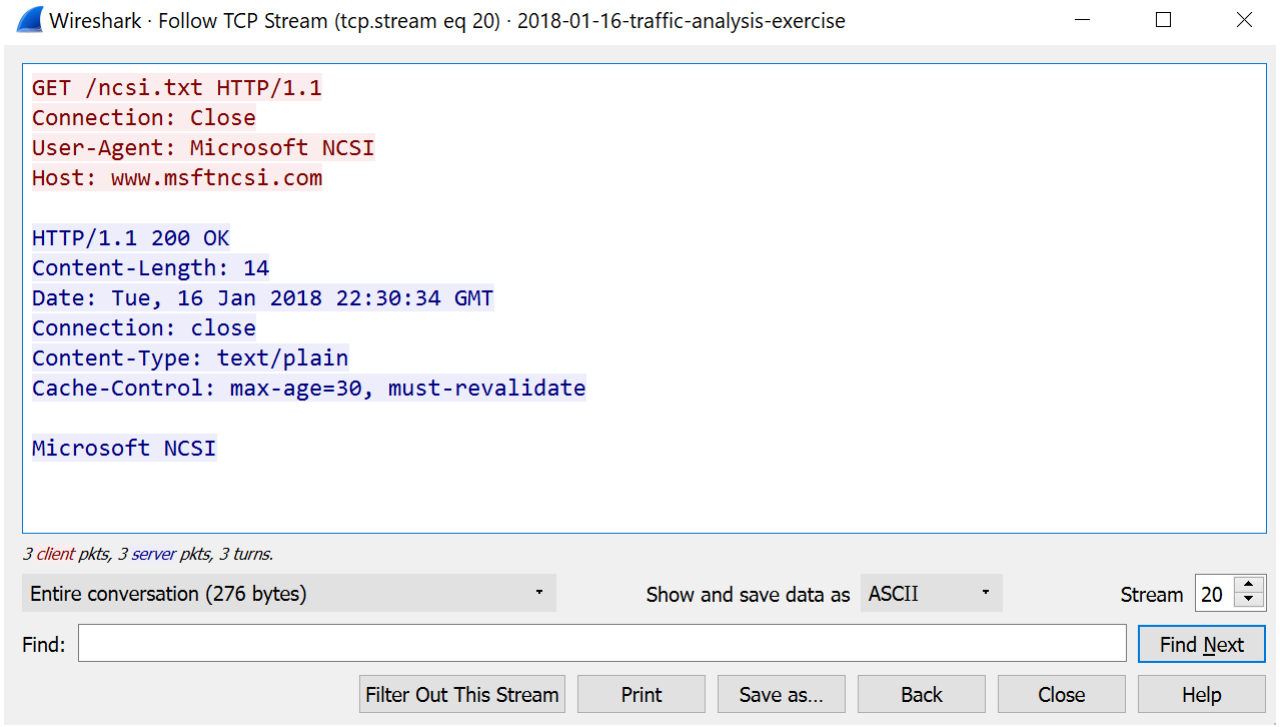
Packet	Hostname	Content Type	Size	Filename
315	www.msftncsi.com	text/plain	14 bytes	ncsi.txt
607	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
622	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
633	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
635	zepter.com	text/html	903 bytes	index.php
643	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
654	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
665	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
668	zepter.com	text/html	706 bytes	index.php
677	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
687	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
700	zepter.com	application/x-www-form-urlencoded	441 bytes	index.php
702	zepter.com	text/html	769 bytes	index.php
712	carfax.com	application/x-www-form-urlencoded	453 bytes	index.php
716	carfax.com	text/html	692 bytes	index.php
718	carfax.com	text/html	692 bytes	index.php

Wireshark Follow TCP Streams

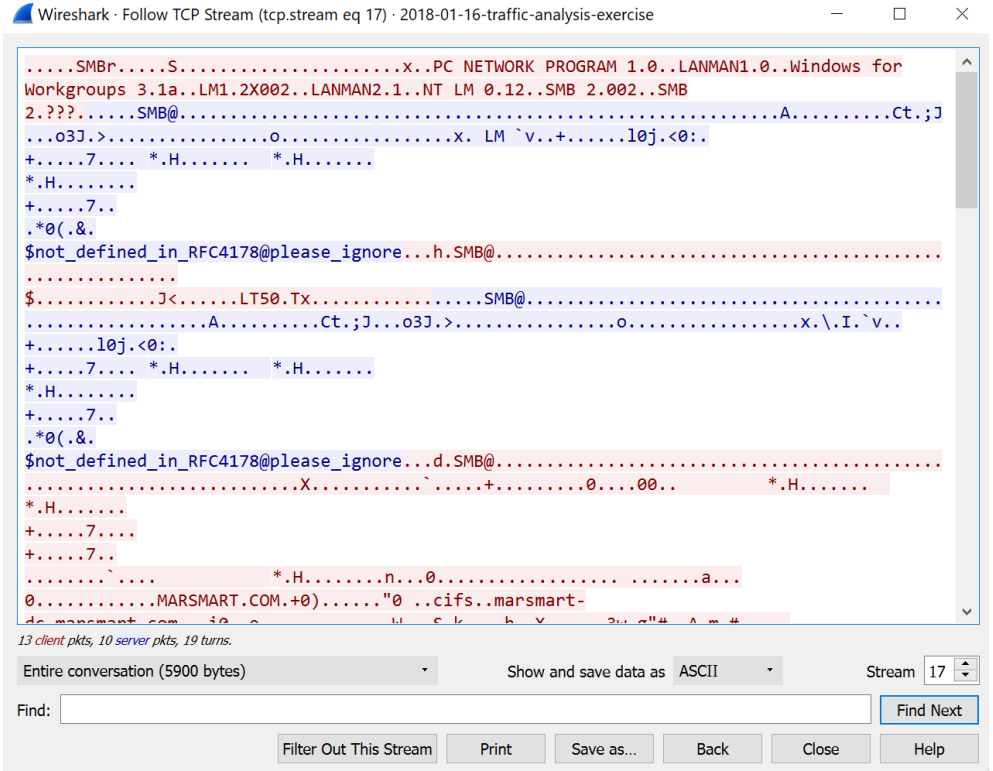
- Limits displayed packets to a specific set
- Can use to export alternate protocols



Followed TCP Stream - HTTP



Followed TCP Stream - SMB



Wireshark Filters

- Capture Filters
  - Use BPF to limit traffic collected
- Display Filters - SLOW
  - Wireshark-specific language
  - Limit traffic displayed
  - Does not alter capture file
  - Intellisense

Wireshark Display Filters

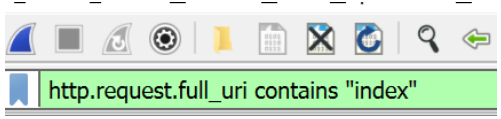
- Equality & comparison
  - Numeric : ==, !=, <, <>, >, <=, >=
  - String : eq, ne, lt, gt, le, ge
  - contains
  - matches
- Logical operators
  - &&, ||, !
  - and, or, not
- Parenthetical notation
  - Use parenthesis to group logic statements

Common Wireshark Display Filters

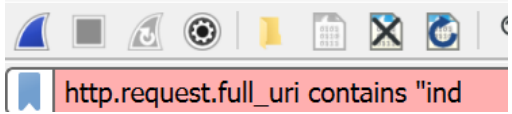
- ip.addr == *host*
- ip.src == *host*
- http.request.full_uri contains “*string*”
- ip.addr == *host* && (dns || http.request.full_uri || ssl.handshake.certificate)
- tcp.port == *port*
- !(ip.addr == *host*)

Display Filters Interface

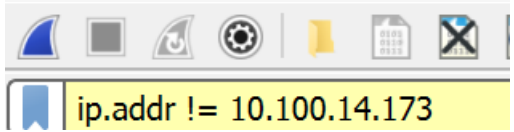
- Working filter



- Broken filter



- Insane filter



TShark

- Command line interface to Wireshark
- Supports display filters
- Far faster
- Supports file exports (recent versions)
- Parse with text-based tools
- Slower than tcpdump, but more functional

TShark Command Line Switches

-r : read pcap file

-nn : do not resolve ports and hosts

-Y : display filter

-T : format output

-e : specify fields

-w : write packets to file

--export-objects : export objects of specified protocol to specified directory

TShark Command Line Switches

```
rangercha@kali:/tmp$ tshark -r 2018-01-16-traffic-analysis-exercise.pcap -nn -Y "dns" -T fields -e "dns.qry.name"
_ldap._tcp.dc._msdcs.marsmart.com
_ldap._tcp.dc._msdcs.marsmart.com
_ldap._tcp.dc._msdcs.marsmart.com
_ldap._tcp.dc._msdcs.marsmart.com
marsmart-dc.marsmart.com
marsmart-dc.marsmart.com
marsmart-dc.marsmart.com
marsmart-dc.marsmart.com
_ldap._tcp.Default-First-Site-Name._sites.marsmart.com
_ldap._tcp.Default-First-Site-Name._sites.marsmart.com
isatap.marsmart.com
isatap.marsmart.com
_ldap._tcp.Default-First-Site-Name._sites.Marsmart-DC.marsmart.com
_ldap._tcp.Default-First-Site-Name._sites.Marsmart-DC.marsmart.com
```

TShark Command Line Switches

```
rangercha@kali:/tmp$ tshark -r 2018-01-16-traffic-analysis-exercise.pcap -nn --export-objects http,export
    1   0.000000 10.100.14.173 → 10.100.14.22 DNS 93 Standard query 0x5475 SRV _ldap._tcp.dc._msdcs.marsmart
    2   0.000179 10.100.14.22 → 10.100.14.173 DNS 153 Standard query response 0x5475 SRV _ldap._tcp.dc._msdcs
.com A 10.100.14.22
```

```
rangercha@kali:/tmp$ ls export/
'index(10).php'  'index(12).php'  'index(14).php'  'index(2).php'  'index(4).php'  'index(6).php'  'index(8).php'   index.php
'index(11).php'  'index(13).php'  'index(1).php'   'index(3).php'  'index(5).php'  'index(7).php'  'index(9).php'   ncsi.txt
rangercha@kali:/tmp$
```

TShark Output Fields

- Separate multiple fields into separate –e arguments
  - tshark -r filename.pcap -T fields **-e 'http.request.full_uri' -e 'http.referer'** -Y 'tcp.port==80’
- I would not call it complete…
  - Some fields are blank for no reason
- List of all possible fields
  - https://www.wireshark.org/docs/dfref/
  - icmp.data
  - http.request.full_uri
  - http.referer

Ngrep

- Find packets matching a regex
- grep, but for packets
- Why not use grep?
  - Need to know the packet information
- Command line switches
  -ql : suppress hash marks, read from file

Ngrep

```
rangercha@kali:/tmp$ ngrep -qI 2018-01-16-traffic-analysis-exercise.pcap 'in.ex'
input: 2018-01-16-traffic-analysis-exercise.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: in.ex

T 10.100.14.173:49193 -> 37.143.160.70:80 [AP] #606
  POST /l5yhus/index.php HTTP/1.1..Cache-Control: no-cache..Connection: Keep-Alive..Pra
  t: zepter.com..User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
  R 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)..Content-Length: 441....
```

Text Parsing

- Find, massage, and present text
- One of the best skills to have
- Not just for packet analysis
- Linux-based tools
  - Also have windows versions

Output Redirection

- stdin, stdout, stderr
- | : direct stdout to stdin
- > : direct stdout to file
- >> : direct stdout append to file
- < : direct file to stdin
- 2>&1 : direct stderr to stdout

Cut

- Splits input into columns
- Outputs selected columns
- Bad for reordering
- Splits on arbitrary characters
- Beware of repeating characters
- Switches
  -f *field_number* : select specified fields
  -d ‘*delimiter’* : use specified delimiter to split

Cut

```
rangercha@kali:/tmp$ tshark -r 2018-01-16-traffic-analysis-exercise.pcap -nn --export-objects http,export -Y 'http' 2>&1 | cut -f 7 -d' '
10.100.14.173
23.204.108.56
37.143.160.70
10.100.14.173
195.228.41.2
10.100.14.173
5.6.73.146
→
81.12.175.59
```

Why does this happen?

Cut command

Translate - tr

- Replaces single character
- tr ‘a’ ‘b’ : replace a with b
- tr -s ‘ ‘ : squeeze repeating characters
- tr -d ‘:‘ : delete character
- Very handy before cut

Tr with Cut

```
rangercha@kali:/tmp$ tshark -r 2018-01-16-traffic-analysis-exercise.pcap -nn --export-objects http,export -Y 'http' 2>&1 | tr -s ' ' | cut -f4 -d' '
10.100.14.173
23.204.108.56
10.100.14.173
37.143.160.70
10.100.14.173
195.228.41.2
10.100.14.173
5.56.73.146
10.100.14.173
81.12.175.59
10.100.14.173
109.96.148.33
10.100.14.173
```

Cut Multiple Columns

```
rangercha@kali:/tmp$ sudo tcpdump -nn -r 2018-01-16-traffic-analysis-exercise.pcap | tr -s ' ' | cut -f5,7-8,10- -d' '
reading from file 2018-01-16-traffic-analysis-exercise.pcap, link-type EN10MB (Ethernet)
10.100.14.22.53: SRV? _ldap._tcp.dc._msdcs.marsmart.com.
10.100.14.173.61280: 1/0/1 SRV 0 100 (111)
10.100.14.22.53: SRV? _ldap._tcp.dc._msdcs.marsmart.com.
10.100.14.173.61280: 1/0/1 SRV 0 100 (111)
10.100.14.22.53: A? marsmart-dc.marsmart.com.
10.100.14.22.53: A? marsmart-dc.marsmart.com.
10.100.14.173.49554: 1/0/0 A (58)
10.100.14.173.49554: 1/0/0 A (58)
10.100.14.22.389: length 170
10.100.14.22.389: length 170
```

cut -f5,7-8,10- -d’ ‘

cut columns 5, 7 through 8, 10, and all columns after 10, delimited by space

Sort

- Orders text
- Alphabetical or numeric
  -n : numeric sort
- Can reverse sort
  -r : reverse sort
- Needed for aggregation

Output Chain

- | tr -s ' ' | cut -f5 -d' ' | cut -f5 -d'.' | tr -d ':' | sort
- Squeeze spaces
- Get column 5, space delimited
- Get column 5, period delimited
- Delete the : character
- Sort output

Unique - uniq

- Aggregates repeating lines
- Can provide a count with –c
- Sort first

Get Top 10 Destination Ports

```
rangercha@kali:/tmp$ sudo tcpdump -nn -r 2018-01-16-traffic-analysis-exercise.pcap
 | tr -s ' ' | cut -f5 -d' ' | cut -f5 -d'.' | tr -d ':' | sort | uniq -c | sort -
rn | head
reading from file 2018-01-16-traffic-analysis-exercise.pcap, link-type EN10MB (Eth
ernet)
    107 88
     83
     71 80
     55 445
     47 389
     36 137
     27 49155
     23 53
     21 5355
     20 49158
```

Head & Tail

- head -n *number*
  - Get top *number* rows (default 10)
- tail -n *number*
  - Get last *number* rows (default 10)

Get Top 10 Destination Ports – New Connections

```
rangercha@kali:/tmp$ sudo tcpdump -nn -r 2018-01-16-traffic-analysis-exercise.pcap
 'tcp[13] = 0x02 '| tr -s ' ' | cut -f5 -d' ' | cut -f5 -d'.' | tr -d ':' | sort |
 uniq -c | sort -rn | head
reading from file 2018-01-16-traffic-analysis-exercise.pcap, link-type EN10MB (Eth
ernet)
     21 88
     14 80
      5 389
      4 445
      2 49155
      2 135
      1 49158
```

Word / Line Count

- wc
- wc -l
  - Count lines
- Answer “how many ports” and “how many hosts” questions

Decoding

- xxd
  - Decode ASCII to hex and hex to ASCII
  - xxd -r -p
    - Hex to ASCII
- Base64
  - Encode/decode base64 encoded content
  - Decode: base64 -d
  - Encode: base64 -e

Practical Exercise - Fragmentation

- frags.pcap
  - Why might only 1 host have responded to the ICMP echo request?
- frags2.pcap
  - What was the server’s response code?
  - What URL was probably requested?

Practical Exercise - Scanning

- nmap.pcap
  - Which host was performing scanning?
  - Which host was the scan target?
  - How many TCP ports were scanned?
  - How many UDP ports were scanned?
  - Which TCP ports were open?
  - What may have caused the scan behavior after 12:04:10?

Practical Exercise – Fragmentation Answers

- frags.pcap
  - Why might only 1 host have responded to the ICMP echo request?
    - Fragmented ICMP blocked by Google, but not by local router
- frags2.pcap
  - What was the server's response code?
    - 301
  - What URL was probably requested?
    - www.google.com

Practical Exercise – Scanning Answers

- nmap.pcap
  - Which host was performing scanning?
    - 192.168.2.175
    - tcpdump -r nmap.pcap -nn 'tcp[13]=0x02' | cut -f3 -d' ' | cut -f1-4 -d '.' | sort | uniq -c | sort -n

```
reading from file nmap.pcap, link-type EN10MB (Ethernet)
      1 192.168.2.45
      3 192.168.2.75
     18 192.168.2.98
   4537 192.168.2.175
```

Practical Exercise – Scanning Answers

- nmap.pcap
  - Which host was the scan target?
    - 192.168.2.75
    - tcpdump -r nmap.pcap -nn 'tcp[13]=0x02' | cut -f5 -d' ' | cut -f1-4 -d '.' | sort | uniq -c | sort –n

```
reading from file nmap.pcap, link-type EN10MB (Ethernet)
      1 65.55.44.109
      4 192.168.2.45
     16 192.168.3.249
   4538 192.168.2.75
```

Practical Exercise – Scanning Answers

- nmap.pcap
  - How many TCP ports were scanned?
    - 1000
    - sudo tcpdump -r nmap.pcap -nn 'tcp[13]=0x02 && src host 192.168.2.175 && dst host 192.168.2.75'| cut -f 5 -d' ' | cut -f5 -d'.'| cut -f1 -d':' | sort | uniq | wc –l
  - How many UDP ports were scanned?
    - 30
    - sudo tcpdump -r nmap.pcap -nn 'udp && src host 192.168.2.175 && dst host 192.168.2.75'| cut -f 5 -d' ' | cut -f5 -d'.'| cut -f1 -d':' | sort | uniq | wc -l

Practical Exercise – Scanning Answers

- nmap.pcap
  - Which TCP ports were open?
    - 1025,1026,1027,1031,1032,1034,135, 139,2869,3306,445,5357
    - sudo tcpdump -r nmap.pcap -nn 'tcp[13]=0x12 && dst host 192.168.2.175 && src host 192.168.2.75'| cut -f 3 -d' ' | cut -f5 -d'.'| cut -f1 -d':' | sort | uniq

```
reading from file nmap.pcap
1025
1026
1027
1031
1032
1034
135
139
2869
3306
445
5357
```

Practical Exercise – Scanning Answers

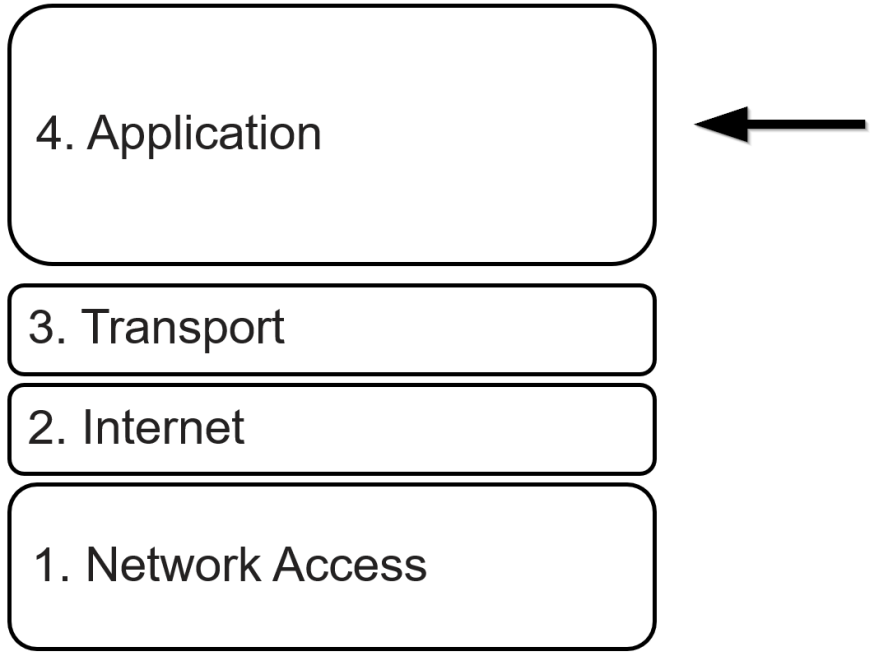
- nmap.pcap
  - What may have caused the scan behavior after 12:04:10?
  - Firewall enabled that does not reset connections

```
56710 → 445 [SYN] Seq=0 Win=29200 L
33664 → 111 [SYN] Seq=0 Win=29200 L
56884 → 256 [SYN] Seq=0 Win=29200 L
33784 → 443 [SYN] Seq=0 Win=29200 L
33602 → 22 [SYN] Seq=0 Win=29200 Le
42012 → 53 [SYN] Seq=0 Win=29200 Le
36270 → 113 [SYN] Seq=0 Win=29200 L
34480 → 23 [SYN] Seq=0 Win=29200 Le
52224 → 587 [SYN] Seq=0 Win=29200 L
51006 → 8080 [SYN] Seq=0 Win=29200
51008 → 8080 [SYN] Seq=0 Win=29200
52230 → 587 [SYN] Seq=0 Win=29200 L
```

Part 3 – Common Protocols

Overview
Part 3 – Common Protocols

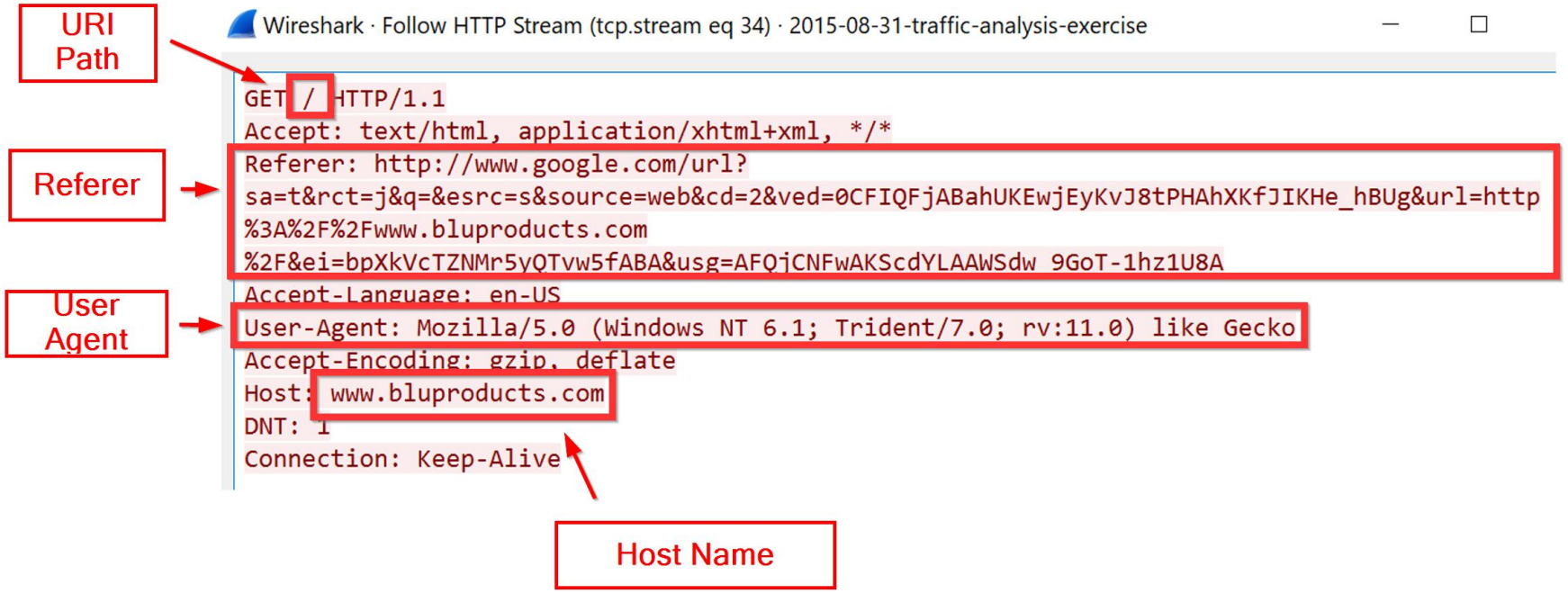
- HTTP
- SSL
- ARP (not app layer)
- DNS
- FTP



HTTP

- Extremely common
- Source of most infections
- Often permitted through firewalls
- Use 2015-08-31-traffic-analysis-exercise.pcap for examples (from malware-traffic-analysis.net)

HTTP Request



HTTP Response

```
HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Mon, 31 Aug 2015 17:56:48 GMT
Expires: Wed, 30 Sep 2015 17:56:48 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 219
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

Important HTTP Response Headers

- “Referer”
  - Tells which page caused the current request
  - May be blank if referred from HTTP to HTTPS
- Location
  - 30X responses send user to “Location” header value
- Content-Type
  - Tells how to interpret server response

HTTP Redirection

- Only certain ways to load content
  - 30X response code
  - iframe
  - script
  - meta refresh
- http://ranger-cha.blogspot.com/2015/02/following-trail-http-referer-header.html

HTTP Forensics

- Starting from an alert…

```
08/31-17:58:18.615707  [**] [1:2021338:10] ET CURRENT_EVENTS Possible Evil Redirecto
Trojan was detected] [Priority: 1] {TCP} 64.20.39.203:80 -> 192.168.137.239:49259
08/31-17:58:18.958675  [**] [1:2025105:2] ET INFO DNS Query for Suspicious .ga Domai
] {UDP} 192.168.137.239:64920 -> 192.168.137.1:53
08/31-17:58:31.536056  [**] [1:2020716:2] ET POLICY Possible External IP Lookup ipin
tion] [Priority: 1] {TCP} 192.168.137.239:49286 -> 54.164.11.220:80
08/31-17:58:33.321082  [**] [1:2022284:3] ET TROJAN AlphaCrypt CnC Beacon 5 [**] [Cl
CP} 192.168.137.239:49287 -> 72.55.148.19:80
08/31-17:58:34.271246  [**] [1:2021724:5] ET TROJAN Alphacrypt/TeslaCrypt Ransomware
as detected] [Priority: 1] {TCP} 72.55.148.19:80 -> 192.168.137.239:49287
08/31-17:59:15.095598  [**] [1:2022284:3] ET TROJAN AlphaCrypt CnC Beacon 5 [**] [Cl
CP} 192.168.137.239:49288 -> 72.55.148.19:80
```

HTTP Forensics

- Use source port

tcp.port == 49259

No.	Time	Source	Destination	Protocol	Length	Info
7639	2015-08-31 …	192.168.137.239	64.20.39.203	TCP	66	49259 → 80 [SYN] S
7643	2015-08-31 …	64.20.39.203	192.168.137.239	TCP	66	80 → 49259 [SYN, A
7644	2015-08-31 …	192.168.137.239	64.20.39.203	TCP	60	49259 → 80 [ACK] S
7645	2015-08-31 …	192.168.137.239	64.20.39.203	HTTP	381	GET / HTTP/1.1
7651	2015-08-31 …	64.20.39.203	192.168.137.239	TCP	54	80 → 49259 [ACK] S
7654	2015-08-31 …	64.20.39.203	192.168.137.239	TCP	1421	80 → 49259 [ACK] S
7655	2015-08-31 …	192.168.137.239	64.20.39.203	TCP	60	49259 → 80 [ACK] S
7656	2015-08-31 …	64.20.39.203	192.168.137.239	TCP	1421	80 → 49259 [ACK] S
7657	2015-08-31 …	64.20.39.203	192.168.137.239	TCP	1421	80 → 49259 [ACK] S

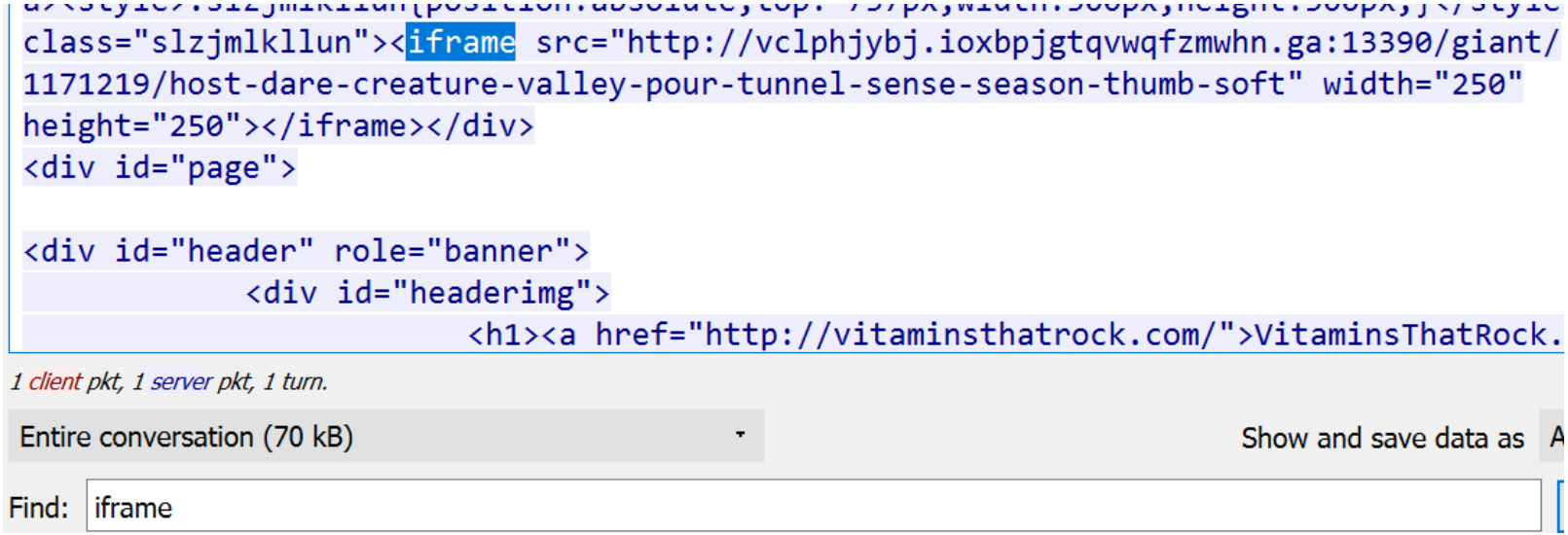
HTTP Forensics

- Follow TCP stream

```
GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://channels.feeddigest.com/domain?d=vitaminsthatrock.com
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
DNT: 1
Connection: Keep-Alive
Host: vitaminsthatrock.com
```

HTTP Forensics

- Find iframe



HTTP Forensics

```
GET /giant/1171219/host-dare-creature-valley-pour-tunnel-sense-season-thumb-soft HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://vitaminsthatrock.com/
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: vclphjybj.ioxbpjgtqvwqfzmwhn.ga:13390
DNT: 1
Connection: Keep-Alive
```

```
    <param value="always" name="allowScriptAccess" />
    <embed name="qvuhndngs" width="167" pluginspage="http://www.macromedia.com/go/
getflashplayer" allowScriptAccess="sameDomain" height="799" type="application/x-shockwave-
flash" quality="high" src="/defense/eGxyaGM" play="true" loop="false" align="middle"/>
</object>

</body>
```

HTTP Forensics

```
</html>GET /defense/eGxyaGM HTTP/1.1
Accept: */*
Accept-Language: en-US
Referer: http://vclphjybj.ioxbpjgtqvwqfzmwhn.ga:13390/giant/1171219/host-dare-creature-
valley-pour-tunnel-sense-season-thumb-soft
x-flash-version: 18,0,0,203
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: vclphjybj.ioxbpjgtqvwqfzmwhn.ga:13390
DNT: 1
Connection: Keep-Alive
```

HTTP Forensics

- After finding the exploit, work backwards from the alert

HTTP Forensics

```
GET /external/http%3A%2F%2Fvitaminsthatrock.com HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://channels.feeddigest.com/domain?d=vitaminsthatrock.com
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: channels.feeddigest.com
DNT: 1
Connection: Keep-Alive
Cookie: _ga=GA1.2.734754304.1441043872; _gat=1; _ym_visorc_31433973=w

HTTP/1.1 302 Found
Server: nginx/1.8.0
Date: Mon, 31 Aug 2015 17:58:15 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Location: http://vitaminsthatrock.com
```

HTTP Forensics

- What happened?
  - User went to google
  - Clicked on a link
  - web.feeddigest.com
  - 302 redirect to vitaminsthatrock.com
  - iframe to vclphjybj.ioxbpjgtqvwqfzmwhn.ga:13390
  - Loaded flash exploit

HTTP Forensics

- Why bother?
  - Effective blocks
  - Supply malware analysis teams
  - Effective threat intelligence
  - Build statistics
  - Notify compromised sites

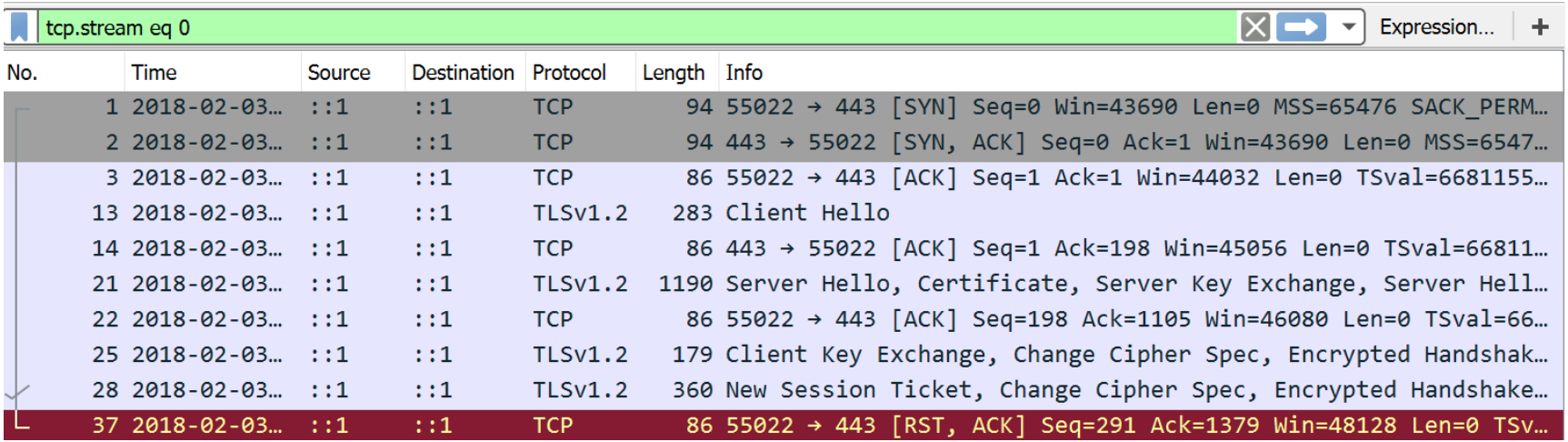
SSL

- Becoming more common
- Uses public key infrastructure (PKI)
- Encrypts other protocols (often HTTP)
- Can authenticate the client, but not common
- Often permitted through firewalls
- ssl_pcap_example.pcap

SSL Connection Steps

- TCP handshake
- Client hello (supported protocols & cipher suites)
- Server cert & key exchange (selected protocol & cipher suite)
- Client session key & change cipher spec

SSL Handshake in Wireshark



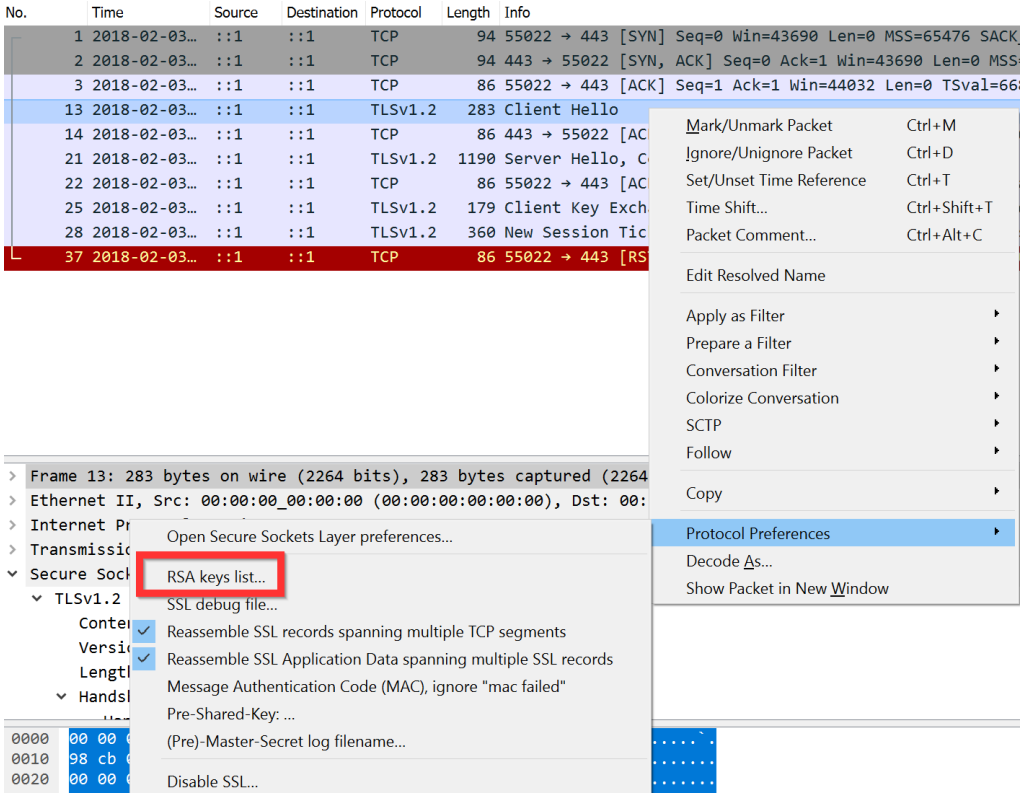
SSL – What do we know?

- Client supported protocols and cipher suites
  - Client public key if used
- Server selected protocol and cipher suite
- Server public certificate
  - May have site name in cert

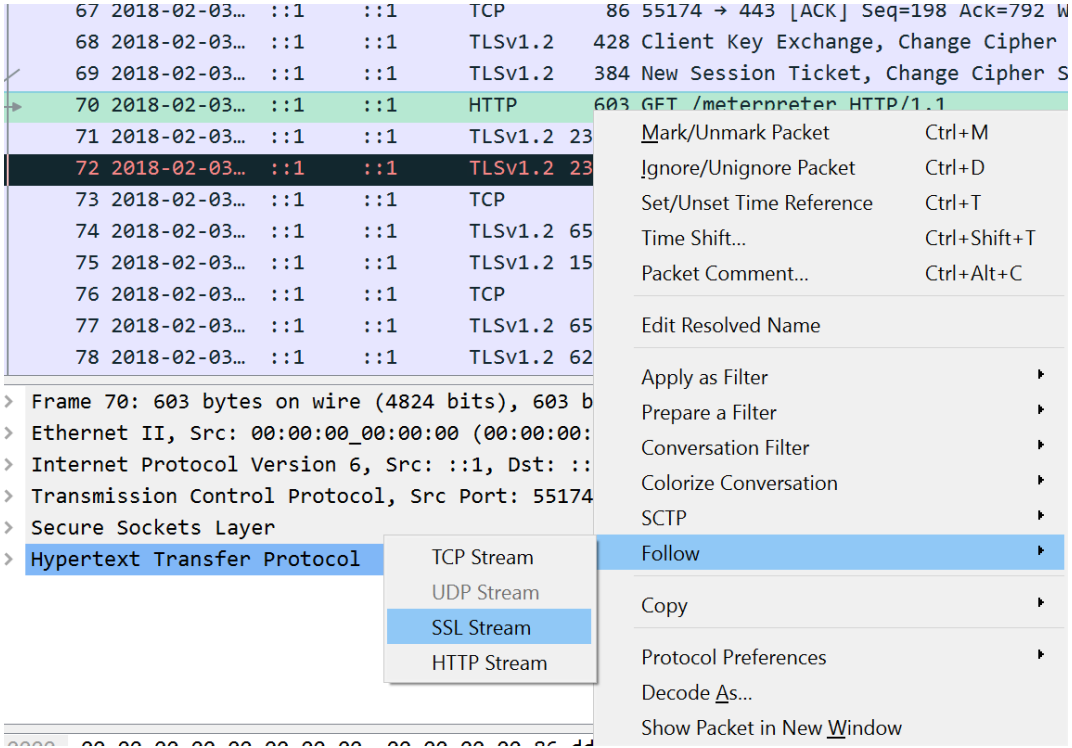
SSL Decryption

- Need to decrypt the session key
- Does not work with Diffie-Hellman key exchange
- Client encryption info is ephemeral
  - Can enable SSL key logging in browser
- Server private key can decrypt

SSL Decryption



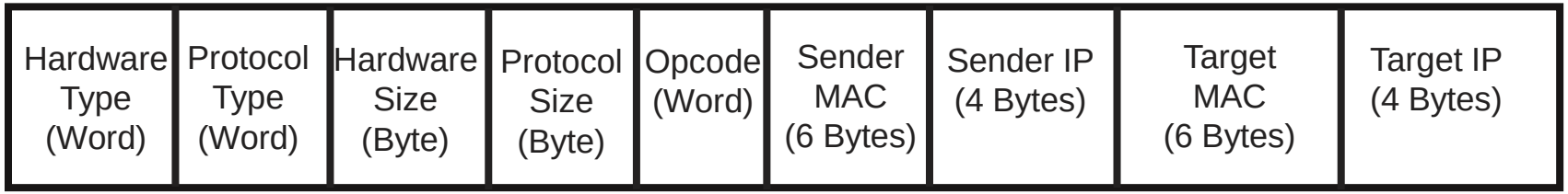
SSL Decryption



Address Resolution Protocol (ARP)

- Resolve MAC address to IP address
- Stays within broadcast domain
- Request and response packets
- No security
  - Allows MitM

Address Resolution Protocol (ARP) Format




```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Apple_a4:3b:c4 (6c:94:f8:a4:3b:c4)
    Sender IP address: 192.168.2.158
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.2.1
```

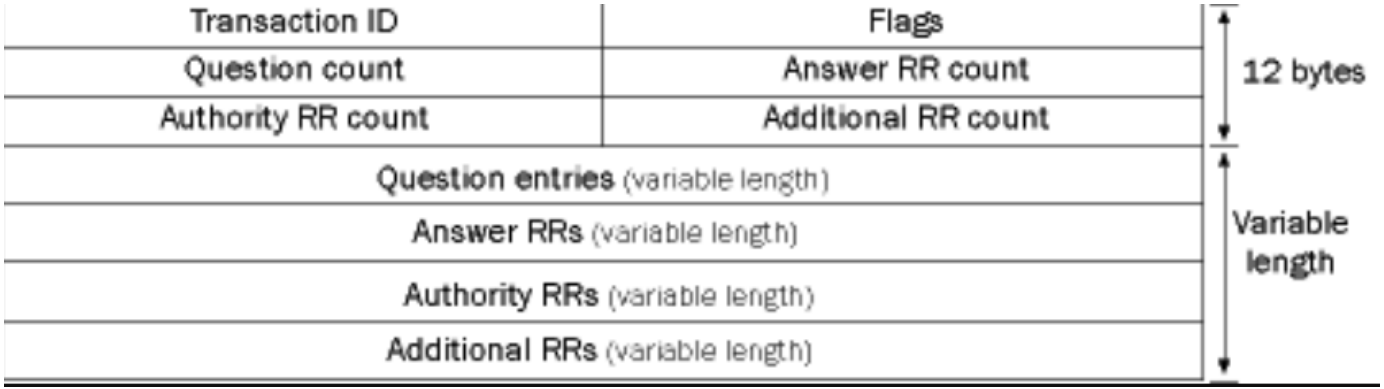
Address Resolution Protocol (ARP)

```
    24 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.244? Tell 192.168.2.175
    25 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.243? Tell 192.168.2.175
    26 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.242? Tell 192.168.2.175
    27 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.241? Tell 192.168.2.175
    28 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.229? Tell 192.168.2.175
    29 2018-02-03…  Vmware_28:20:8c     Broadcast           ARP         42 Who has 192.168.2.36? Tell 192.168.2.175
    30 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.45 is at 00:0c:29:28:20:8c
    31 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.60 is at 00:0c:29:28:20:8c
    32 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.60 is at 00:0c:29:28:20:8c
    33 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.60 is at 00:0c:29:28:20:8c
    34 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.75 is at 00:0c:29:28:20:8c
    35 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.79 is at 00:0c:29:28:20:8c
    36 2018-02-03…  Vmware_28:20:8c     Raspberr_72:a5:28   ARP         60 192.168.2.110 is at 00:0c:29:28:20:8c
```

Domain Name Service (DNS)

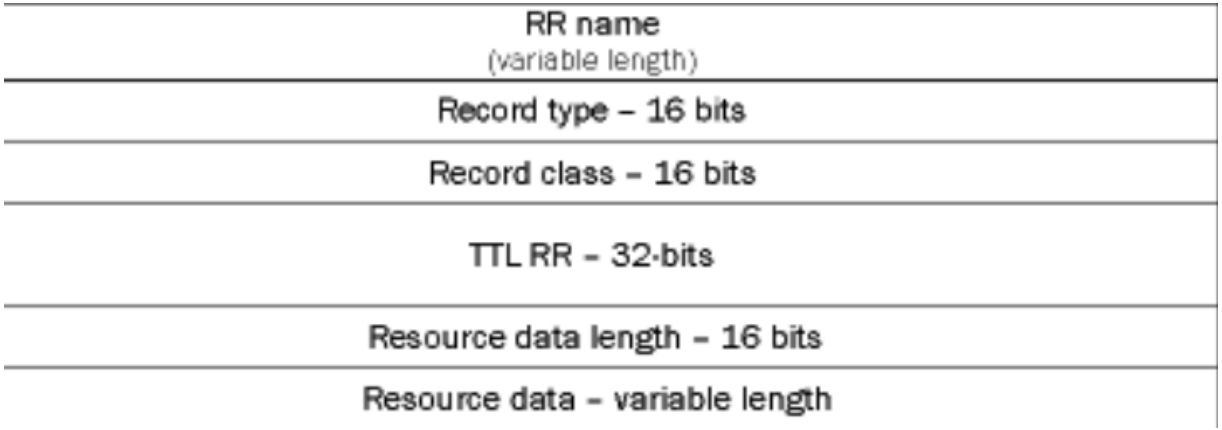
- Resolves host names to IP addresses
- Often occurs before malicious traffic
- Also used for data exfiltration
- Usually UDP 53
  - Can be TCP 53

DNS Query Header



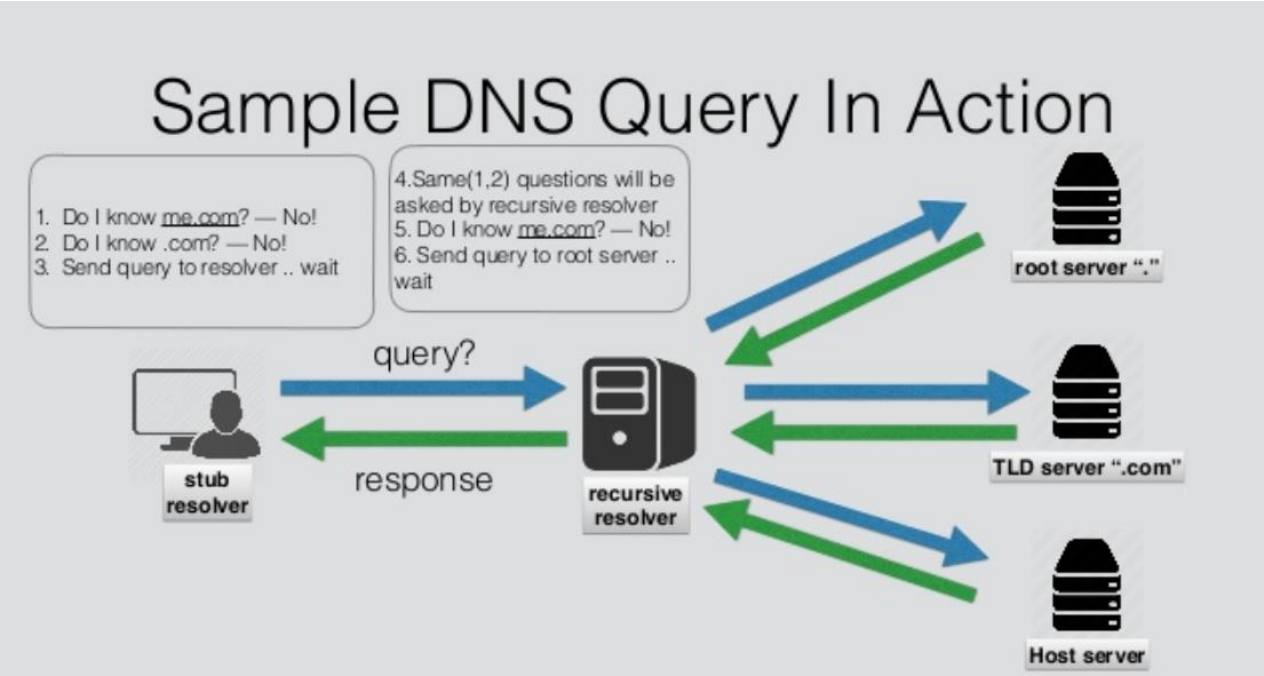
https://technet.microsoft.com/en-us/library/bb962025.aspx

DNS Response Header



https://technet.microsoft.com/en-us/library/bb962025.aspx

DNS Recursion



DNS Odd Behavior

	Time	Source	Destination	Protocol	Length	Info
524	2015-11-30…	52.2.…	10.42.…	DNS	156	Standard query response 0xe3d3 TXT cmd.sg1.atnascorp.com TXT
571	2015-11-30…	10.42…	52.2.2…	DNS	131	Standard query 0xe3d4 TXT cmd.sg1.atnascorp.com
573	2015-11-30…	52.2.…	10.42.…	DNS	188	Standard query response 0xe3d4 TXT cmd.sg1.atnascorp.com TXT
574	2015-11-30…	10.42…	52.2.2…	DNS	170	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
575	2015-11-30…	10.42…	52.2.2…	DNS	190	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
576	2015-11-30…	10.42…	52.2.2…	DNS	218	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
577	2015-11-30…	10.42…	52.2.2…	DNS	194	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
578	2015-11-30…	10.42…	52.2.2…	DNS	222	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
579	2015-11-30…	10.42…	52.2.2…	DNS	230	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
580	2015-11-30…	10.42…	52.2.2…	DNS	206	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
581	2015-11-30…	10.42…	52.2.2…	DNS	198	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
582	2015-11-30…	10.42…	52.2.2…	DNS	250	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
583	2015-11-30…	10.42…	52.2.2…	DNS	226	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
584	2015-11-30…	10.42…	52.2.2…	DNS	242	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
585	2015-11-30…	10.42…	52.2.2…	DNS	198	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT
586	2015-11-30…	10.42…	52.2.2…	DNS	218	Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT

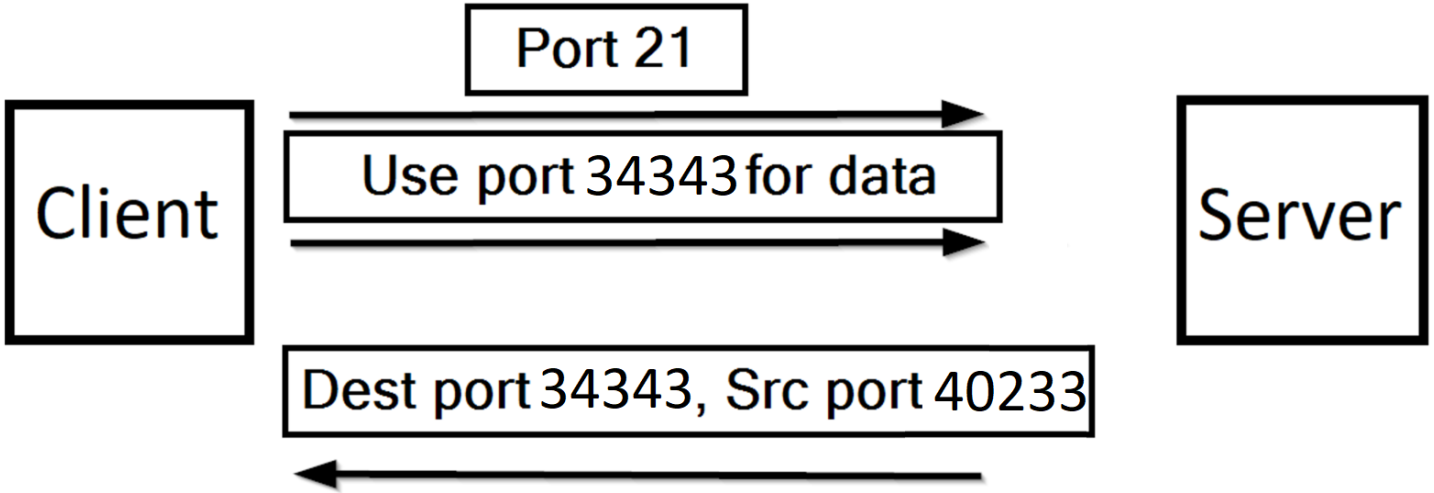
File Transfer Protocol

- Plaintext protocol for transferring files
- May be wrapped in SSL (FTPS)
- Not to be confused with SFTP (SSH file transfer)
- USER command to send username
- PASS command to send password
- Two types of FTP, active and passive

Active Mode FTP

- Client connects to server on port 21 (command channel)
- **Client gives server** an ephemeral port for data (data channel)
- **Server connects back to client** on data channel (**ephemeral source and destination**)

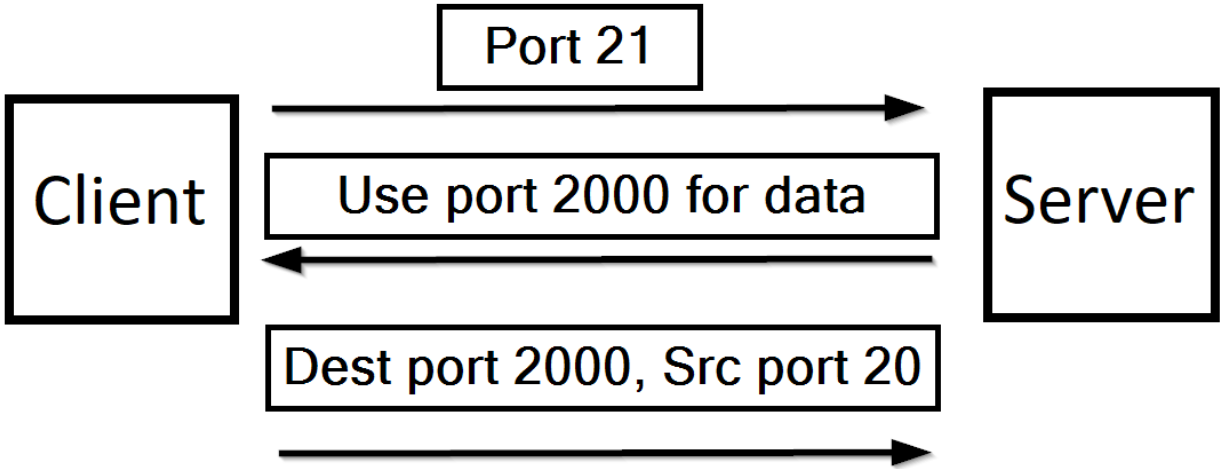
Active Mode FTP



Passive Mode FTP

- Client connects to server on port 21 (command channel)
- **Server gives client** an ephemeral port for data (data channel)
- **Client connects to server** on data channel (source port 20)

Passive Mode FTP



Part 4 – Scapy

Scapy

- Python packet crafting library
- Written by Philippe Biondi
- Slow, but simple
  - Not great for broad scanning
- Helpful for simulating odd behavior
- Allows complex sequences of packets

Scapy

- Define each layer of a packet
- Ping localhost
  - testpacket = IP(dst="127.0.0.1")/ICMP()
- Connect to Bing on port 80 TCP
  - testpacket = IP(dst=”www.bing.com”)/TCP(dport=80)/”GET / HTTP/1.1\n\n”
  - Not going to work great
    - Data on SYN
    - Networking stack will reset the connection

Scapy – useful commands

- ls(): list all object members

```
>>> ls(IP)
version    : BitField (4 bits)                   = (4)
ihl        : BitField (4 bits)                   = (None)
tos        : XByteField                          = (0)
len        : ShortField                          = (None)
id         : ShortField                          = (1)
flags      : FlagsField (3 bits)                 = (0)
frag       : BitField (13 bits)                  = (0)
ttl        : ByteField                           = (64)
proto      : ByteEnumField                       = (0)
chksum     : XShortField                         = (None)
src        : SourceIPField (Emph)                = (None)
dst        : DestIPField (Emph)                  = (None)
options    : PacketListField                     = ([])
```

Scapy – useful commands

- object.summary: list non-default field values

```
>>> testpacket.summary
<bound method IP.summary of <IP  frag=0 proto=tcp
 dst=Net('www.bing.com') |<TCP  dport=http |<Raw
 load='GET / HTTP/1.1\n\n' |>>>>
```

Scapy – useful commands

- Resp, noresp = sr(testpacket)
  - Send packet, match responses. Store responses in resp, unanswered packets in noresp
- sr1(testpacket)
  - Send packet, match single response
- send(testpacket)
  - Send packet, do not look for responses

Scapy – notes

- Native networking stack may get confused
  - Diagnose with tcpdump
  - Use iptables to prevent it
  - iptables --save >> iptables_original
  - iptables –A OUTPUT –p tcp –tcp-flags RST RST –j DROP
    - Fix using iptables --restore iptables_original or iptables --flush
- Run Scapy as root

Part 5 – Analysis Tips

Common External Ports

- Good egress filtering should block most external traffic
- Permitted traffic should go through an intermediary
  - TCP:80 (HTTP)
  - TCP:443 (SSL)
  - UDP:123 (NTP)
    - Should be blocked
  - UDP:53 (DNS)

Common Internal TCP Ports

- 22 (SSH)
- 445 (SMB)
- 88 (Kerberos)
- 135 (DCE/RPC)
- 389 (LDAP)
- 636 (LDAPS)
- 993 (IMAPS)

Common Internal TCP Ports

- 80 (HTTP)
- 443 (HTTPS)
- 8080 (Alternate HTTP)
- 8443 (Alternate HTTPS)
- Ephemeral ports (RPC)

Common Internal UDP Ports

- 53 (DNS)
- 5355(LLMNR)
- 123(NTP)
- 514(SYSLOG)

Alerts

- Many sources
  - Intrusion Detection System
  - Intrusion Prevention System
  - Web Application Firewall
- Signatures are not always great
- Places to start
  - Use the source port

Continued Analysis

- Work forward for post infection
  - Find binary files for analysis
  - Identify command and control traffic
- Work backward to find the origin
  - Often starts with legitimate sites

Useful Techniques

- Wireshark display filters
  - dns || http.request.full_uri || ssl.handshake.certificate
- Find “odd” URLs
  - Long alphanumeric strings that are not words
  - Directed outside of domain
  - Use “referer” to work backwards
  - Find redirection call in calling page (URL string)

Automated Tools

- Virustotal.com
  - Binaries or pcaps
- Sandboxes
  - https://zeltser.com/automated-malware-analysis/

More Training

- https://malware.dontneedcoffee.com/
- https://contagiodump.blogspot.com/
- https://www.malware-traffic-analysis.net/
- https://www.root-me.org/

Part 6 – Practical Exercise

ch2.pcap
(from root-me.org)

- Find the FTP password

ch3.pcap
(from root-me.org)

- Find the twitter password

2015-03-24-traffic-analysis-exercise.pcap (from malware-traffic-analysis.net)

- IP address of the infected host
- Which server is probably legitimate, but compromised?
- What redirection techniques does this exploit kit use?
- What malware was used?

sansholidayhack2013.pcap (from the SANS holiday hack challenge, 2013)

- What are the MACs and IPs of the machines that ARP cache poison? Is it successful?
- What are the IPs of the systems that were port scanning?
- What systems and protocols did the scan discover?
- What account was used in an attack over the SMB protocol?

packet_intro.pcap

- What are the max and minimum SSL/TLS versions supported by 192.168.2.122?
- Find the 7 flags

Part 7 – Practical Exercise Answers

ch2.pcap

- Find the FTP password
  - ngrep for PASS

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ ngrep -qI ch2.pcap 'PASS'
input: ch2.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: PASS

T 10.20.144.150:35974 -> 10.20.144.151:21 [AP] #11
  PASS cdts3500..
```

ch3.pcap

- Find the twitter password
- ngrep -qI ch3.pcap 'Authorization: Basic' | tr ':' '\n' | grep 'Basic' | cut -f3 -d' ' | cut -f1 -d'.' | base64 –d

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ ngrep -qI ch
3.pcap 'Authorization: Basic' | tr ':' '\n' | grep 'Basic' | cut -f3 -d' '
 | cut -f1 -d'.' | base64 -d
usertest:passwordrangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_l
ong$
```

2015-03-24-traffic-analysis-exercise.pcap

- IP address of the infected host
  - 192.168.122.200
- Which server is probably legitimate, but compromised?
  - forums.pelicanparts.com
- What redirection techniques does this exploit kit use?
  - Script tag injection
- What types of exploits were served?
  - Adobe Reader, Flash, Java, Silverlight

sansholidayhack2013.pcap

- What are the MACs and IPs of the machines that ARP cache poison? Is it successful?
  - 00:0c:29:f7:f4:9a - 10.21.22.253 - successful
  - 10.25.22.252 - unsuccessful
- What are the IPs of the systems that were port scanning?
  - 10.25.22.252,10.21.22.253,10.25.22.253
  - tcpdump -nn -r sansholidayhack2013.pcap 'tcp[13]=0x02' | cut -f3,5 -d' ' | tr ' ' '.' | cut -f1-4,6-10 -d'.' | sort | uniq -c | sort -rn
  - tcpdump -nn -r sansholidayhack2013.pcap 'tcp[13]=0x14' | cut -f5 -d' ' | cut -f1-4 -d'.' | sort | uniq -c | sort -n

sansholidayhack2013.pcap

- What systems and protocols did the scans discover?
  - tcpdump -nn -r sansholidayhack2013.pcap 'tcp[13]=0x12 && (dst host 10.25.22.252 || dst host 10.21.22.253 || dst host 10.25.22.253)' | cut -f3 -d' ' | sort | uniq
- What account was used in an attack over the SMB protocol?

sansholidayhack2013.pcap

10.16.11.5.110	10.25.22.22.80	165.254.158.56.80	208.80.154.234.80
10.21.22.22.502	10.25.22.23.80	173.194.43.47.443	208.80.154.240.80
10.21.22.23.502	10.25.22.250.80	192.190.173.45.80	216.22.25.175.80
10.21.22.24.502	10.25.22.30.80	192.204.3.75.80	54.230.49.239.80
10.21.22.253.1225	10.25.22.58.4444	199.7.57.72.80	63.245.217.36.80
10.2.2.2.8081	10.25.22.58.445	208.80.154.224.80	69.16.175.10.80
72.167.239.239.80	74.125.226.239.443	82.103.140.42.443	74.125.226.228.80

sansholidayhack2013.pcap

72.21.195.198.443	74.125.226.242.443	10.25.22.22.44818
72.21.203.211.80	74.125.226.251.80	165.254.138.136.80
72.21.214.3.443	74.125.22.82.80	208.80.154.225.80
72.21.215.52.80	81.169.180.37.443	69.195.141.178.443
74.125.226.199.443	81.169.180.37.80	82.103.134.102.80

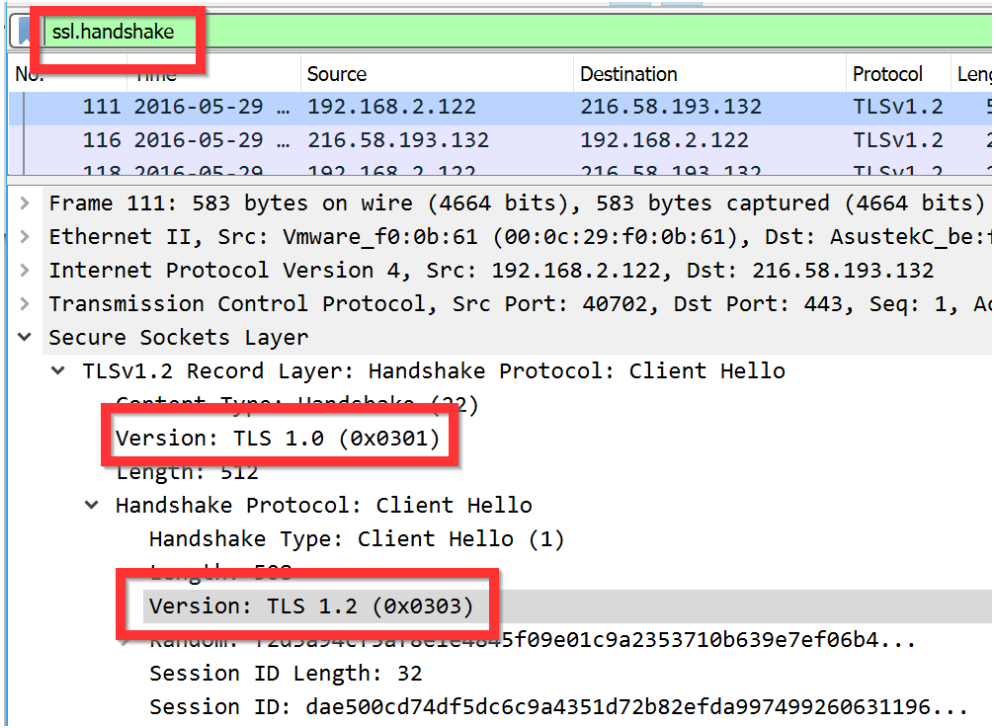
sansholidayhack2013.pcap

- What account was used in an attack over the SMB protocol?
  - ernie

packet_intro.pcap

- What are the max and minimum SSL/TLS versions supported by 192.168.2.122?
  - Max: TLS 1.2
  - Min: TLS 1.0

packet_intro.pcap



packet_intro.pcap

- Flag 1
  - http.request.full_uri contains “flag”
- Flag 2

```
root@kali:/mnt/hgfs/vmshared/training/packet_intro_long# ngrep -qI packet_intro.pcap
'flag2'
input: packet_intro.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: flag2

T 192.168.2.122:57196 -> 66.235.120.113:80 [AP] #2344
  GET / HTTP/1.1..Host: askjeevs.com..User-Agent: flag2:82d5927b53538c2da5c4e5eadba2b
f2a..Accept: */*....
root@kali:/mnt/hgfs/vmshared/training/packet_intro_long#
```

packet_intro.pcap

- Flag 2
  - ngrep -qI packet_intro.pcap ‘flag2’

```
root@kali:/mnt/hgfs/vmshared/training/packet_intro_long# ngrep -qI packet_intro.pcap
'flag2'
input: packet_intro.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: flag2

T 192.168.2.122:57196 -> 66.235.120.113:80 [AP] #2344
  GET / HTTP/1.1..Host: askjeevs.com..User-Agent: flag2:82d5927b53538c2da5c4e5eadba2b
f2a..Accept: */*....
root@kali:/mnt/hgfs/vmshared/training/packet_intro_long#
```

packet_intro.pcap

- Flag 3
  - ngrep -qI packet_intro.pcap 'flag3'

```
root@kali:/mnt/hgfs/vmshared/training/packet_intro_long# ngrep -qI packet_intro.pcap
'flag3'
input: packet_intro.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: flag3

U 192.168.2.122:60663 -> 8.8.8.8:53 #4750
  .V.......... b768e1e8075fd6c1b7c11c84536dd467.flag3.com.....

U 8.8.8.8:53 -> 192.168.2.122:60663 #4757
  .V.......... b768e1e8075fd6c1b7c11c84536dd467.flag3.com.....-.........?.dns1.re
  gistrar-servers.com,,..hostmaster.Mx..0.........:.....
```

packet_intro.pcap

- Flag 4
  - echo "flag" | base64
  - ngrep -qI packet_intro.pcap 'ZmxhZ’
  - ngrep -qI packet_intro.pcap 'ZmxhZ' | grep 'ZmxhZ' | tr -s ' ' | cut -f3 -d' ' | cut -f2- -d'=' | base64 -d

packet_intro.pcap

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ echo "flag" | base64
ZmxhZwo=
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ ngrep -qI packet_intro.pcap 'ZmxhZ'
input: packet_intro.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: ZmxhZ

T 192.168.2.122:56154 -> 174.36.107.130:80 [AP] #5040
  GET /?a=ZmxhZzQ6MDIxMDFkZDNmNWYxZjA0NGRiYjM1YTUzMDUyMjQxOTc== HTTP/1.1..Host: supersketch.com..Use
r-Agent: curl/7.47.
  0..Accept: */*....

rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ ngrep -qI packet_intro.pcap 'ZmxhZ' |
grep 'ZmxhZ' | tr -s ' ' | cut -f3 -d' ' | cut -f2- -d'=' | base64 -d
flag4:02101dd3f5f1f044dbb35a5305224197base64: invalid input
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$
```

packet_intro.pcap

- Flag 5
  - echo "flag5" | xxd -ps
  - ngrep -qI packet_intro.pcap '666c616735’
  - printf "666c6167353a6436386663732316461333165656639326635653739396164666435346531
    39" | xxd -r -p

packet_intro.pcap

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ echo "flag5" | xxd -ps
666c6167350a
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ ngrep -qI packet_intro.pcap '666c61673
5'
input: packet_intro.pcap
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: 666c616735

T 192.168.2.122:51588 -> 204.79.197.200:80 [AP] #5712
  GET /search=666c6167353a643638666337323164613331656566393266356537393961646664353465313
  9.1..Host: bing.com..User-Agent: curl/7.47.0..Accept: */*....

T 204.79.197.200:80 -> 192.168.2.122:51588 [AP] #5728
  HTTP/1.1 301 Moved Permanently..Location: http://www.bing.com/search=666c6167353a64363866633732
  3164613331656566393266356537393961646664353465313
  9..Server: Microsoft-IIS/8.5..X-MSEdge-Ref: Re
  f A: FE730BDA1B4241D7AC872F3CE666680F Ref B: 36FB7548E3AF536AF11F30FF82345B4A Ref C: Sun May 29
   19:15:33 2016 PST..Date: Mon, 30 May 2016 02:15:32 GMT..Content-Length: 0....
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ printf "666c6167353a643638666337323164
6133316565663932663565373939616466643534653139" | xxd -r -p
```

packet_intro.pcap

- Flag 6
  - tshark -r packet_intro.pcap -T fields -e icmp.type -Y 'icmp' | xargs printf '%x' | xxd -r -p

packet_intro.pcap

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ tshark -r
packet_intro.pcap -T fields -e icmp.type -Y 'icmp' | xargs printf '%x'
666c6167363a363935373464306161633937323061343238373632376337373130323839
3330ddddddddddddddddddddddddddddddddddddddddrangercha@kali:/mnt/hgfs/vmsha
red/training/packet_intro_long$
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ tshark -r
packet_intro.pcap -T fields -e icmp.type -Y 'icmp' | xargs printf '%x' |
 xxd -r -p
flag6:69574d0aac9720a4287627c771028930□□□□□□□□□□□□□□□□□□□□□□rangercha@kali:
/mnt/hgfs/vmshared/training/packet_intro_long$
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$
```

packet_intro.pcap

- Flag 7
  - tcpdump -r packet_intro.pcap -nnA 'icmp' | grep '\.\.' | cut -f4 -d'-' | grep -v '@\.\.\.\.\.\.z' | tr -d '\n'

packet_intro.pcap

```
rangercha@kali:/mnt/hgfs/vmshared/training/packet_intro_long$ sudo tcpdump
 -r packet_intro.pcap -nnA  'icmp' | grep '\.\.' | cut -f4 -d'-'   | grep
-v '@\.\.\.\.\.\.z' | tr -d '\n'
[sudo] password for rangercha:
reading from file packet_intro.pcap, link-type EN10MB (Ethernet)
flag7:02ae9641b7052bfa4124dc41943cf36crangercha@kali:/mnt/hgfs/vmshared/tr
```

DIRECTDEFENSE

www.directdefense.com