Packet Capture and Analysis tcpdump & wireshark

Acknowledgement

- Original slides prepared by
  - Fakrul Alam
  - Yoshinobu Matsuzaki

Why we need to capture packets & how it is related to security?

- Hosts send and receive packets to communicate with each other
- When you have network related issues, you sometimes need to investigate the actual packets to find the root cause
- Also, it helps you to understand behavior of protocols and devices

Networking is layering

- We are seeing only the top surface of the network
- Many hidden actions are ongoing underneath
  - HTTP is not the only protocol you use when browsing web pages
- Troubles may be hidden deep under the protocols

Do you know…

- What kind of protocols are used to get a web page to your browser?
  - Of course Hyper Text Transfer Protocol (HTTP)
  - and what else?

Protocols are invisible

- You must have tools to make them visible
- You must know how each protocol works to detect something weird

tcpdump

- tcpdump is a utility to capture and analyze packets
  - https://www.tcpdump.org/
  - You can watch network interfaces to capture and analyze packets
  - You can save a series of packets captured to a file for later analysis
- Contents of packets are decoded and shown on screen
  - But you still need to know how each protocol works ☺
- You have many options to filter packets to focus on the parts you are interested in

libpcap

- You may be interested in the libpcap library, developed by the same team as that of tcpdump developers
  - https://www.tcpdump.org/
- You can integrate the powerful packet capture and filtering functions with your own program
- There are many language binding modules

What you can see

- Packets sent from or received by your network interface
  - Traffic related to your node
  - Broadcasted packets
  - Multicasted packets to the group which your node joins
- You cannot see traffic that you are not involved in

Basics

```
# tcpdump -nn -i eth0
# tcpdump -nn -i eth0 host 10.0.255.1
# tcpdump -nn -i eth0 dst host 10.0.255.1 and tcp
# tcpdump -nn -i eth0 src net 10.0.0.0/16 and tcp and portrange 1-1024
```

-nn	don’t use DNS to resolve IPs and display port no
-i	interface to watch
dst	watch only traffic destined to a net, host or port
src	watch only traffic whose src is a net, host or port
net	specifies network
host	specifies host
port	specifies a port (i.e. 80, http)
portrange	specifies a range of ports
protocol name	i.e. ip, icmp, ip6, icmp6, tcp, udp

More output

```
# tcpdump -nn -i eth0 -v
# tcpdump -nn -i eth0 -vv
# tcpdump -nn -i eth0 -vvv
```

-v, -vv, -vvv display more detailed packet contents information

```
# tcpdump -nn -i eth0 -x
# tcpdump -nn -i eth0 -X
```

-x, -X display raw binary data of packets in HEX format

Capture size

```
# tcpdump -nn -i eth0 -s 100
# tcpdump -nn -i eth0 -c 1000
```

-s limit the length of sample of each packet to specified size
setting to 0 means to capture the entire packet (precisely speaking,
-s0 option limits the max length to 262,144 bytes)

-c quit the program once the specified number of packets are captured

Some complex expressions

```
# tcpdump -nn -i eth0 not port 22 and dst host 10.0.255.1
# tcpdump -nn -i eth0 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0'
# tcpdump -nn -i eht0 icmp[icmptype] != icmp-echo and icmp[icmptype] !=
icmp-echoreply
# tcpdump -nn -i eth0 icmp6 and ip[40] == 133
# tcpdump -nn -i eth0 icmp6[icmp6type] == icmp6-routersolicit
```

not	negate the condition
tcp[tcpflags]	indicate TCP flags
icmp[icmptype]	indicate ICMPv4 type value
ip[OFFSET]	indicate the value at position OFFSET from the beginning of the protocol header
icmp6[icmp6type]	indicate ICMPv6 type value (available only in newer libpcap library)

Save and Load dump files

```
# tcpdump -nn -i eth0 -w dumpfile.pcap -c 1000
# tcpdump -nn -i eth0 -r dumpfile.pcap port http
```

-w specify the output file
-r specify the input file

There are a lot of other options to control the behavior of tcpdump
See the man page for more advanced operation

tcpdump output

```
11:27:41.823896 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [S], seq
606189042, win 16384, options [mss 1460,nop,nop,sackOK], length 0

11:27:41.922650 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [S.], seq
3302128502, ack 606189043, win 16384, options [mss 1452,nop,nop,sackOK],
length 0

11:27:41.922688 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [.], ack 1,
win 17424, length 0

11:27:41.922887 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [P.], seq
1:61, ack 1, win 17424, length 60

11:27:42.023713 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [P.], seq
1:42, ack 61, win 65475, length 41
```

tcpdump output

```
11:27:41.823896 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [S], seq
606189042, win 16384, options [mss 1460,nop,nop,sackOK], length 0

11:27:41.922650 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [S.], seq
3302128502, ack 606189043, win 16384, options [mss 1452,nop,nop,sackOK],
length 0

11:27:41.922688 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [.], ack 1,
win 17424, length 0

11:27:41.922887 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [P.], seq
1:61, ack 1, win 17424, length 60

11:27:42.023713 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [P.], seq
1:42, ack 61, win 65475, length 41
```

tcpdump output

```
11:27:41.823896 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [S], seq
606189042, win 16384, options [mss 1460,nop,nop,sackOK], length 0

11:27:41.922650 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [S.], seq
3302128502, ack 606189043, win 16384, options [mss 1452,nop,nop,sackOK],
length 0

11:27:41.922688 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [.], ack 1,
win 17424, length 0

11:27:41.922887 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [P.], seq
1:61, ack 1, win 17424, length 60

11:27:42.023713 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [P.], seq
1:42, ack 61, win 65475, length 41
```

tcpdump output

```
11:27:41.823896 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [S], seq
606189042, win 16384, options [mss 1460,nop,nop,sackOK], length 0

11:27:41.922650 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [S.], seq
3302128502, ack 606189043, win 16384, options [mss 1452,nop,nop,sackOK],
length 0

11:27:41.922688 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [.], ack 1,
win 17424, length 0

11:27:41.922887 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [P.], seq
1:61, ack 1, win 17424, length 60

11:27:42.023713 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [P.], seq
1:42, ack 61, win 65475, length 41
```

tcpdump output

```
11:27:41.823896 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [S], seq
606189042, win 16384, options [mss 1460,nop,nop,sackOK], length 0

11:27:41.922650 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [S.], seq
3302128502, ack 606189043, win 16384, options [mss 1452,nop,nop,sackOK],
length 0

11:27:41.922688 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [.], ack 1,
win 17424, length 0

11:27:41.922887 IP 192.168.0.114.3331 > 207.46.26.167.1863: Flags [P.], seq
1:61, ack 1, win 17424, length 60

11:27:42.023713 IP 207.46.26.167.1863 > 192.168.0.114.3331: Flags [P.], seq
1:42, ack 61, win 65475, length 41
```

Wireshark

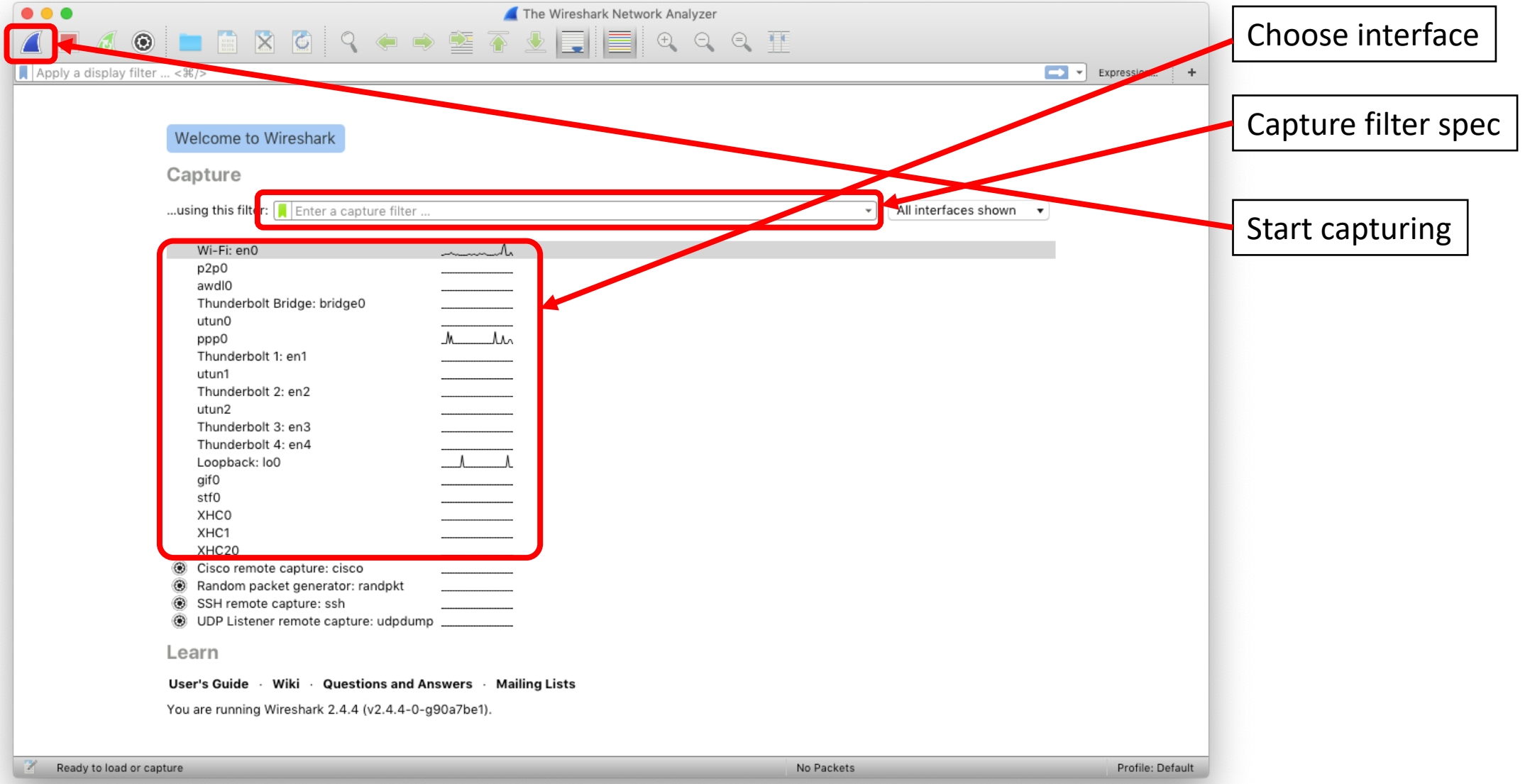
- tcpdump is good enough
- tcpdump is sometimes the only tool we can use especially when using a character terminal
- Wireshark is an opensource GUI application providing similar functions as tcpdump does
  - https://www.wireshark.org/
- It works on Windows, Mac, Linux/UNIX
  - Prebuilt packages for Windows/Mac are available on the Web
  - Many Linux/UNIX systems has their own Wireshark package

Why Wireshark

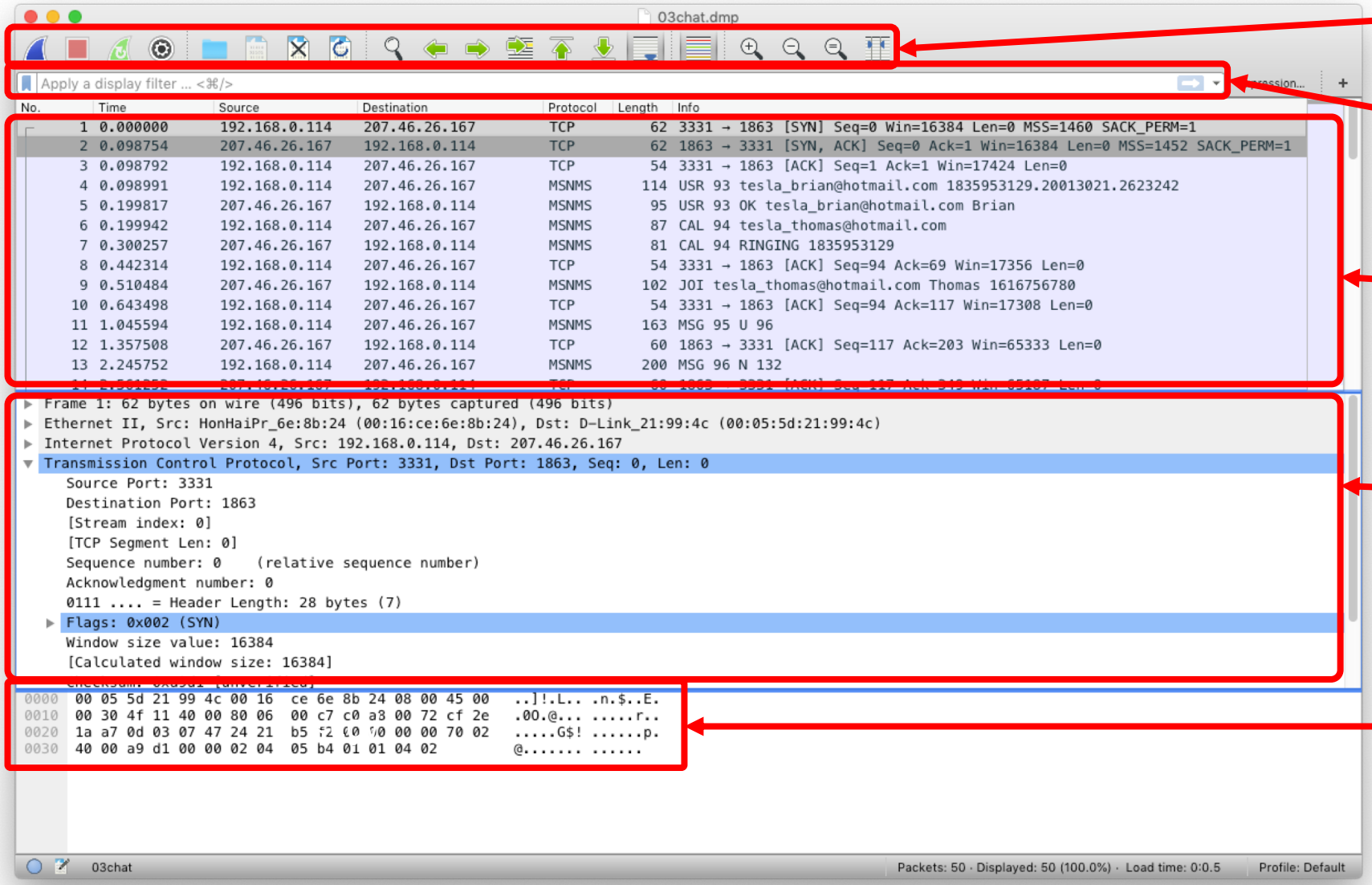
- Network admins use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

- Wireshark is NOT an intrusion detection system
- Wireshark will NOT MANIPURATE things on the network, it only “WATCH” things happening on the network

Capturing


The Wireshark Network Analyzer
Apply a display filter ... <⌘/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Wi-Fi: en0
p2p0
awdl0
Thunderbolt Bridge: bridge0
utun0
ppp0
Thunderbolt 1: en1
utun1
Thunderbolt 2: en2
utun2
Thunderbolt 3: en3
Thunderbolt 4: en4
Loopback: lo0
gif0
stf0
XHC0
XHC1
XHC20
Cisco remote capture: cisco
Random packet generator: randpkt
SSH remote capture: ssh
UDP Listener remote capture: udpdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 2.4.4 (v2.4.4-0-g90a7be1).
Ready to load or capture
No Packets
Profile: Default
Choose interface
Capture filter spec
Start capturing

Window layout



Command & Menu

Display filter spec

Overview of packets

Decoded contents

Raw binary data

Filtering

- Filtering at capturing
- Filtering when displaying

Capture filters

- Filter which packets to capture from interfaces
- Use the same expression as tcpdump filter expression
- Avoid packet loss due to capturing overhead in a heavy traffic condition
- Reduce memory size to keep captured packet data
- Reduce disk size when writing out to a file

Display filter

- Filter which packets to display on screen
- Useful for interactive analysis using the GUI screen
- Use a different filtering language from that of capture filter
  - https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html

Display filter basics

- ip.addr == 10.0.0.1 [Sets a filter for any packet with 10.0.0.1, as either the source or dest]
- ip.addr==10.0.0.1 && ip.addr==10.0.0.2 [sets a conversation filter between the two defined IP addresses]
- http or dns [sets a filter to display all http and dns]
- tcp.port==4000 [sets a filter for any TCP packet with 4000 as a source or dest port]
- tcp.flags.reset==1 [displays all TCP resets]
- http.request [displays all HTTP GET requests]
- tcp contains rviews [displays all TCP packets that contain the word 'rviews'. Excellent when searching on a specific string or user ID]
- !(arp or icmp or dns) [masks out arp, icmp, dns, or whatever other protocols may be background noise. Allowing you to focus on the traffic of interest]

Display filter example

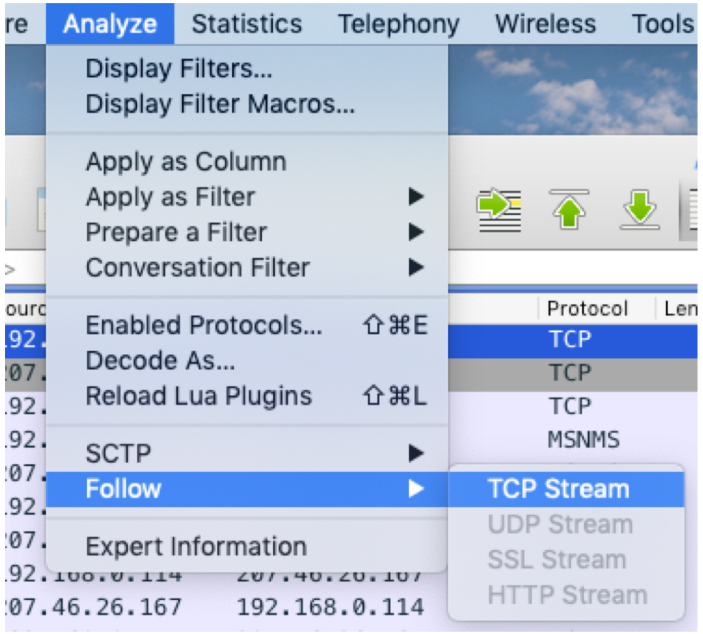
- Showing ICMPv4 packets sent to or received from 10.206.205.105

ip.addr==10.206.205.105 && icmp

No.	Time	Source	Destination	Protocol	Length	Info
225	0.525159	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
417	6.756788	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
659	12.040169	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
675	12.544903	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
923	21.052425	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
930	21.587968	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
1252	32.066055	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)
1275	32.566232	10.206.204.2	10.206.205.105	ICMP	70	Destination unreachable (Port unreachable)

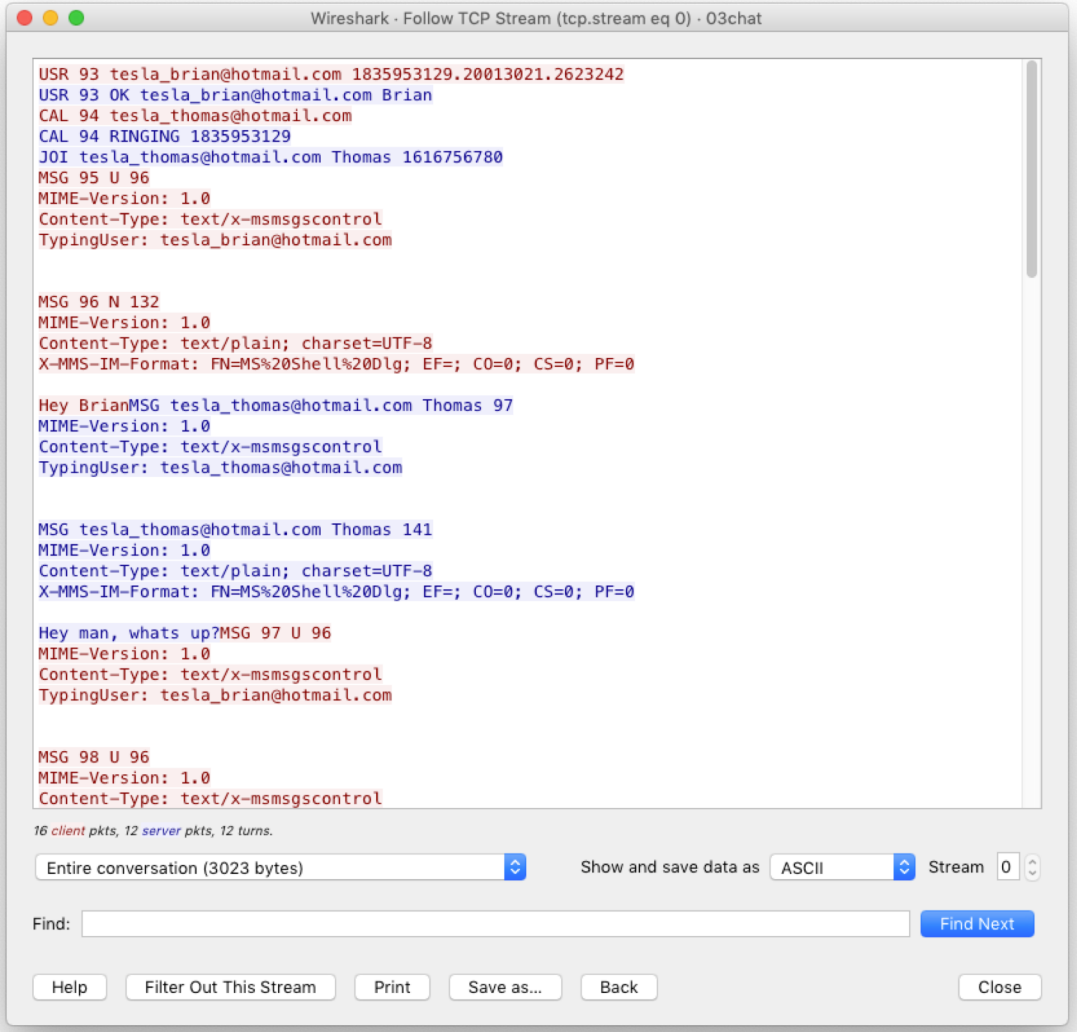
Following TCP stream

- Select a one of the packets of a TCP stream
- Select Analyze->Follow->TCP Stream



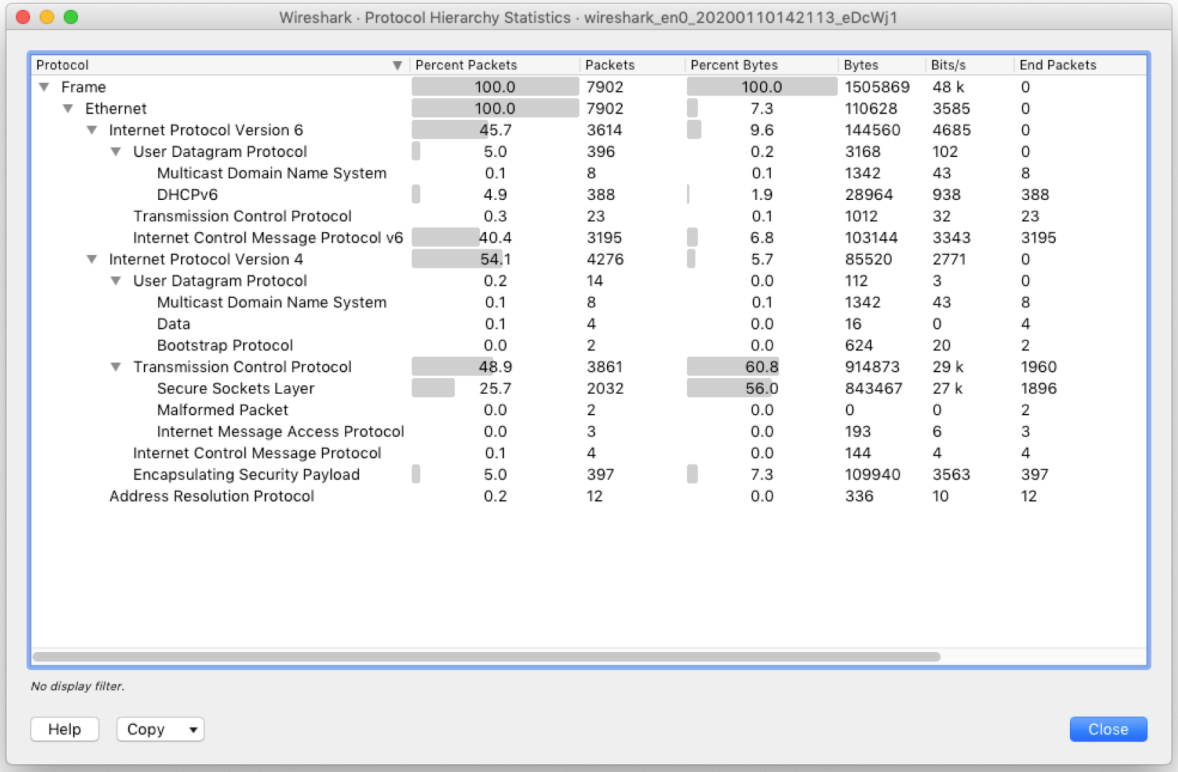
Following TCP stream

- You can reconstruct data exchange of the specified pair of TCP connection
- It is useful as long as the contents are printable



Using Statistics menu

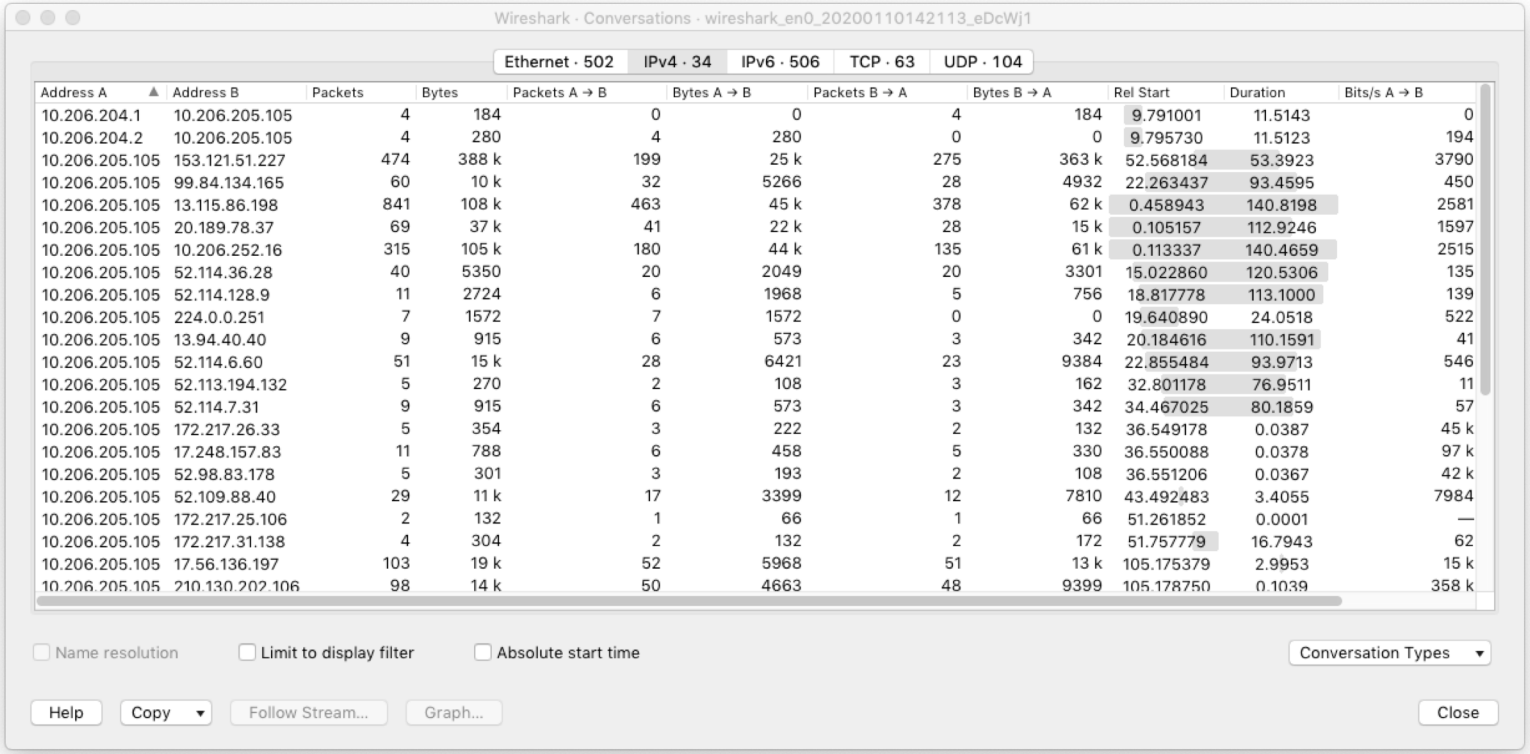
- Brose what kind of protocols are used
  - Statistics->Protocol Hierarchy



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets
Frame	100.0	7902	100.0	1505869	48 k	0
Ethernet	100.0	7902	7.3	110628	3585	0
Internet Protocol Version 6	45.7	3614	9.6	144560	4685	0
User Datagram Protocol	5.0	396	0.2	3168	102	0
Multicast Domain Name System	0.1	8	0.1	1342	43	8
DHCPv6	4.9	388	1.9	28964	938	388
Transmission Control Protocol	0.3	23	0.1	1012	32	23
Internet Control Message Protocol v6	40.4	3195	6.8	103144	3343	3195
Internet Protocol Version 4	54.1	4276	5.7	85520	2771	0
User Datagram Protocol	0.2	14	0.0	112	3	0
Multicast Domain Name System	0.1	8	0.1	1342	43	8
Data	0.1	4	0.0	16	0	4
Bootstrap Protocol	0.0	2	0.0	624	20	2
Transmission Control Protocol	48.9	3861	60.8	914873	29 k	1960
Secure Sockets Layer	25.7	2032	56.0	843467	27 k	1896
Malformed Packet	0.0	2	0.0	0	0	2
Internet Message Access Protocol	0.0	3	0.0	193	6	3
Internet Control Message Protocol	0.1	4	0.0	144	4	4
Encapsulating Security Payload	5.0	397	7.3	109940	3563	397
Address Resolution Protocol	0.2	12	0.0	336	10	12

Using Statistics menu

- Brose who is contacting with whom
  - Statistics->Conversations



tcpdump and Wireshark

- The saved capture file uses the same format
- You can capture packets on remote server without GUI using tcpdump, and analyze it locally using Wireshark GUI

Exercise

- Install Wireshark into your PC
- Run Wireshark and capture inbound/outbound traffic
- Download capture files from
  - http://xxx.xxx.xxx.xxx/WiresharkExerciseData.zip

Exercise 1: Good Old Telnet

- File
  - 01telnet.pcap
- Question: Reconstruct the telnet session
- Q1: Who logged into 192.168.0.1
  - Username _____, Password _____
- Q2: After logged in what did the user do?

Exercise 2: Massive TCP SYN

- File
  - 02massivesyn1.pcap
  - 02massivesyn2.pcap
- Question: Point the difference between them
- Q1: 02massivesyn1.pcap is a _____ attempt
- Q2: 02massivesyn2.pcap is a _____ attempt

Exercise 3: Chatty Employees

- File
  - 03chat.pcap
- Q1: What kind of protocol is used? _____
- Q2: This is a conversation between _____@_____ and _____@_____
- Q3: What do they say about you (sysadmin)?

Exercise 4: Suspicious FTP activity

- File
  - 04ftp1.pcap
- Q1: 10.121.70.151 is FTP _____
- Q2: 10.234.125.254 is FTP _____
- Q3: FTP Err Code 530 means _____
- Q4: 10.234.125.254 attempt _____

Exercise 5: Unidentified traffic

- File
  - 05foobar.pcap
- Q1: See what is going on with Wireshark GUI
  - Statistics->Conversations
    - List->TCP
- Q2: Which application use TCP/6346?

Exercise 6: Covert channel

- File
  - 06convertinfo.pcap
- Question: Take a close look! This is not a typical ICMP Echo/Reply
- Q1: What kind of tool do they use?
- Q2: Name other application which tunnel user traffic

Exercise 7: Analyze Malware

- File
  - 07malware.pcap
- Q1: Find the bad HTTP traffic
- Q2: Is there any malware in the HTTP traffic?
- Q3: Upload one sample malware to https://www.virustotal.com/
  - Does all antivirus detect the malware?

- Tips
  - Use display filter ‘http contains “in DOS mode”’

Exercise 8: SIP

- File
  - 08sip_chat.pcap
- Q1: Can we listen to SIP voice?

Exercise 9: SSH

- Capture your ssh session to your workshop VM
- Q1: Can you monitor what is going on?