Hands-on Network Traffic Analysis

2016 Cyber Defense Boot Camp

What is this module about?

?

What is this module about?

- How to read network traffic?
  1) What does it look like?
  2) How to make sense out of it?

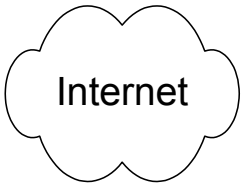
- Prerequisite: network packet & packet analyzer: (header, data)
  – Enveloped letters inside another envelope

Prerequisite: TCP/IP Model (1/9)

- How does Internet work?

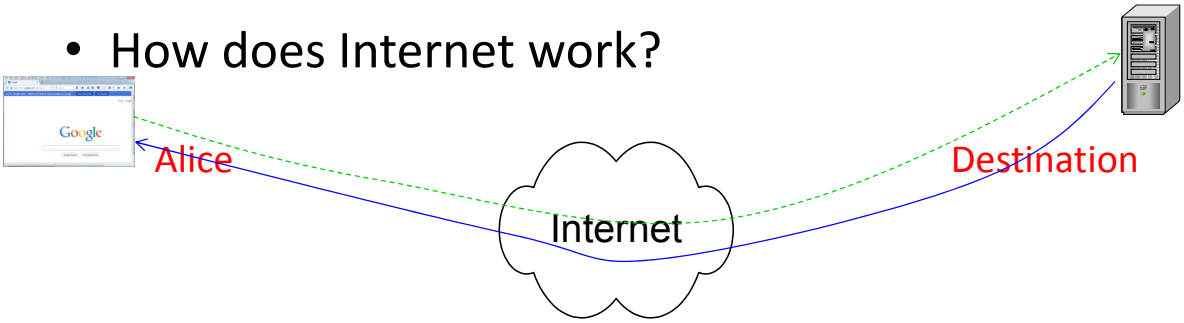
Alice

Destination



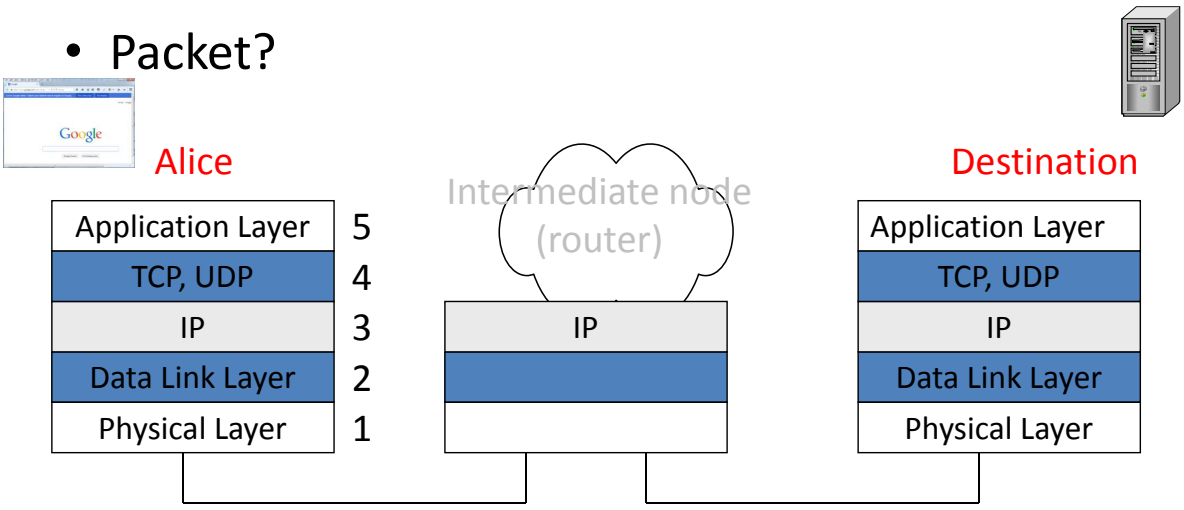
Prerequisite: TCP/IP Model (2/9)

- How does Internet work?



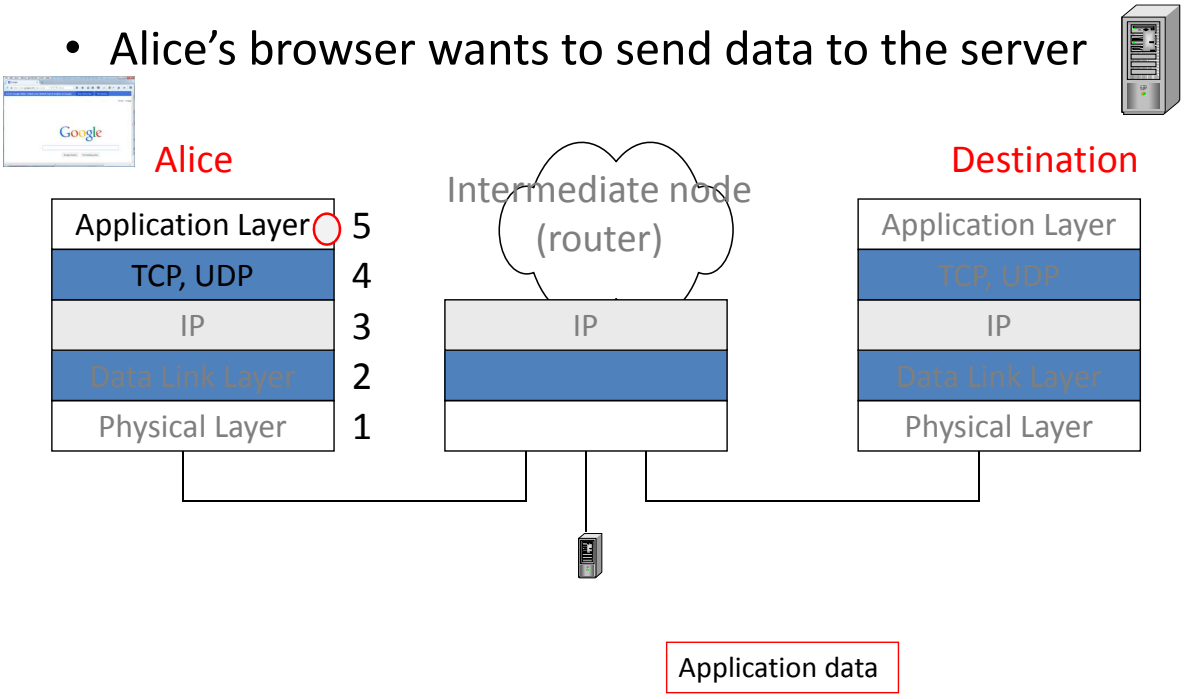
Prerequisite: TCP/IP Model (3/9)

- Packet?



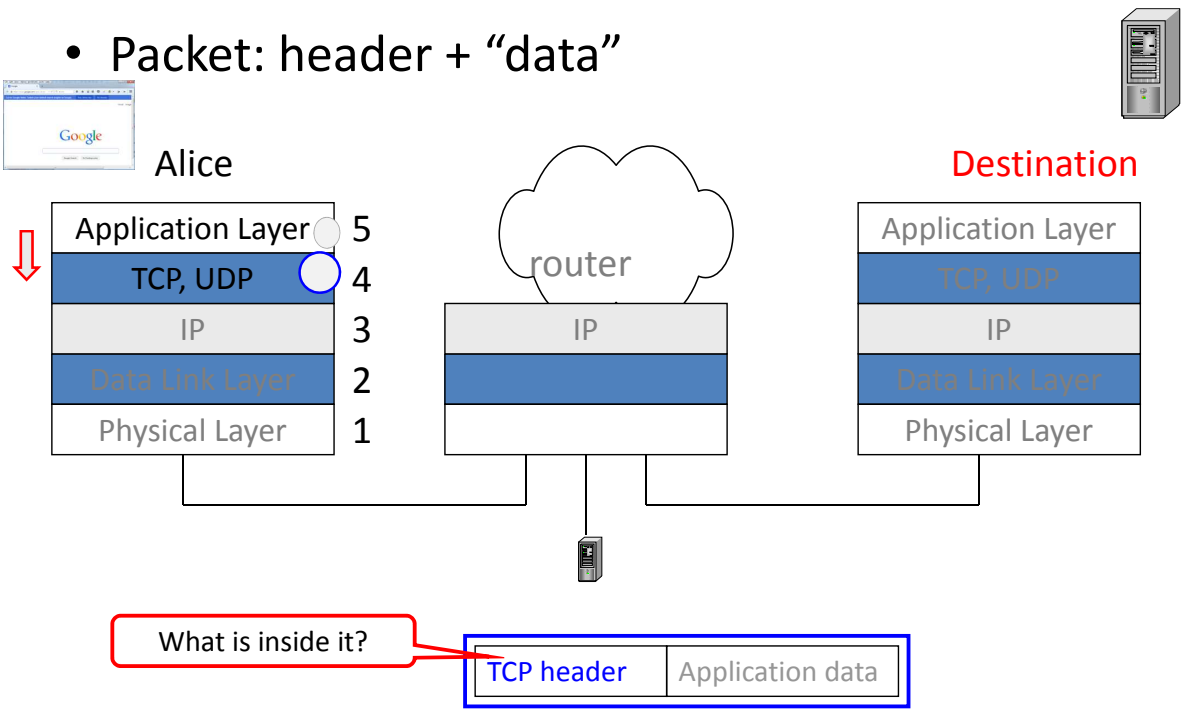
Prerequisite: TCP/IP Model (4/9)

- Alice's browser wants to send data to the server



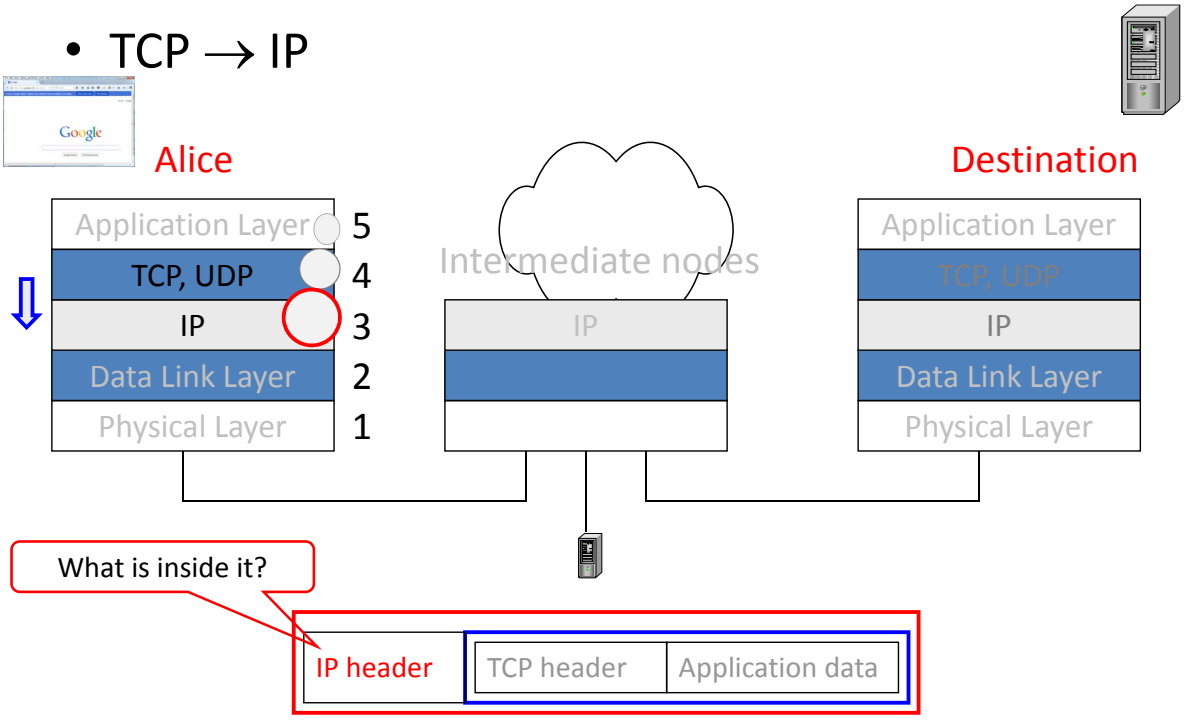
Prerequisite: TCP/IP Model (5/9)

- Packet: header + "data"



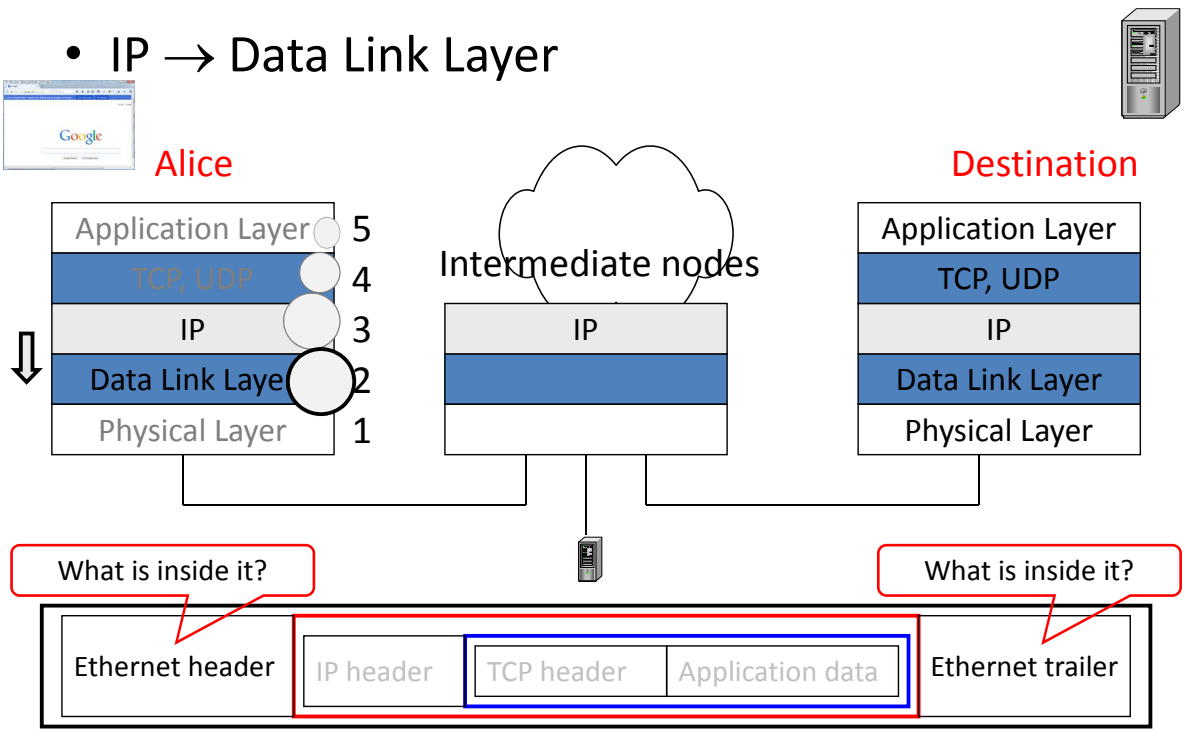
Prerequisite: TCP/IP Model (6/9)

- TCP → IP



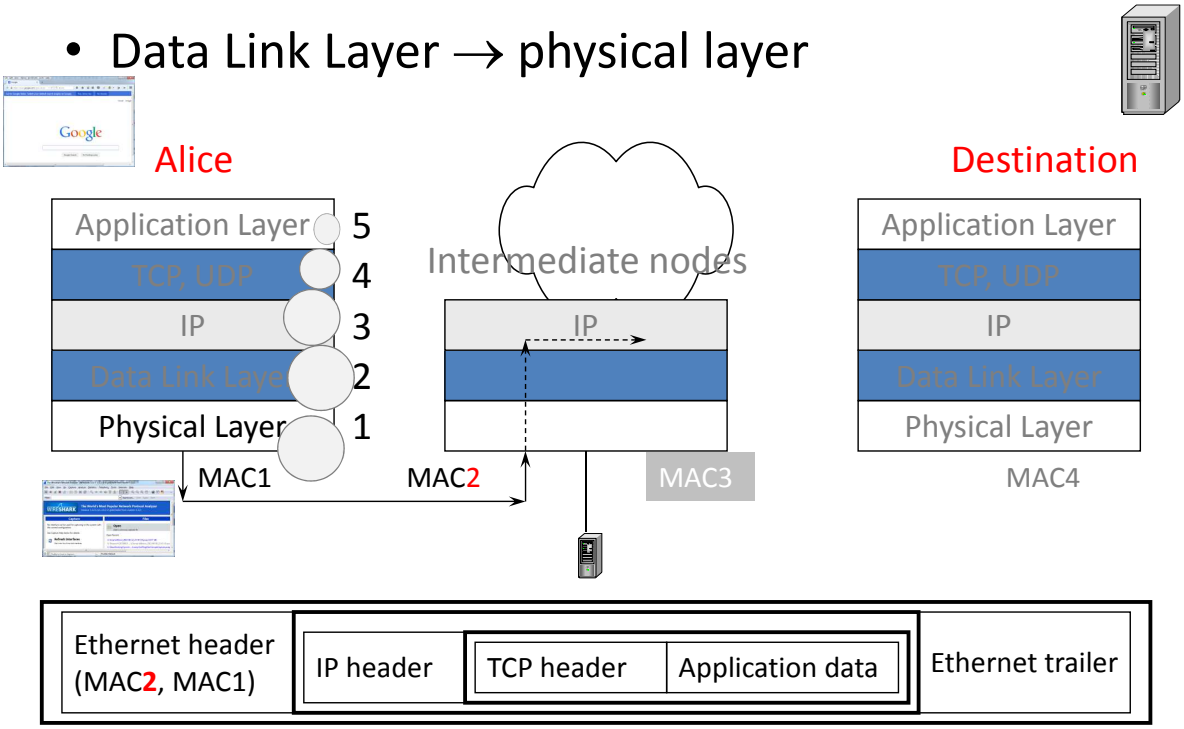
Prerequisite: TCP/IP Model (7/9)

- IP → Data Link Layer



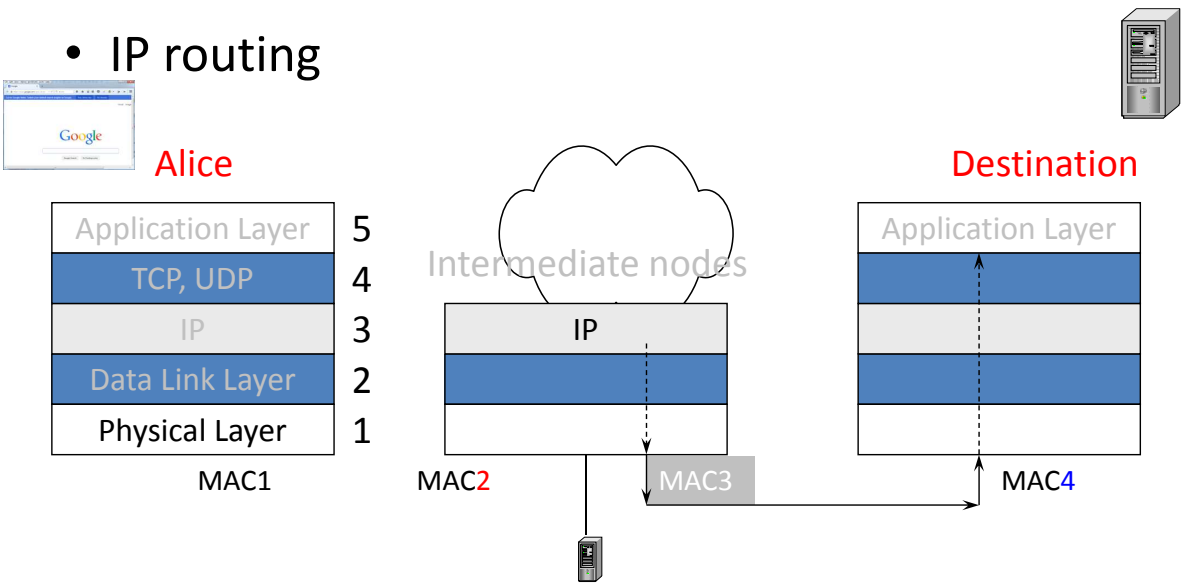
Prerequisite: TCP/IP Model (8/9)

- Data Link Layer → physical layer



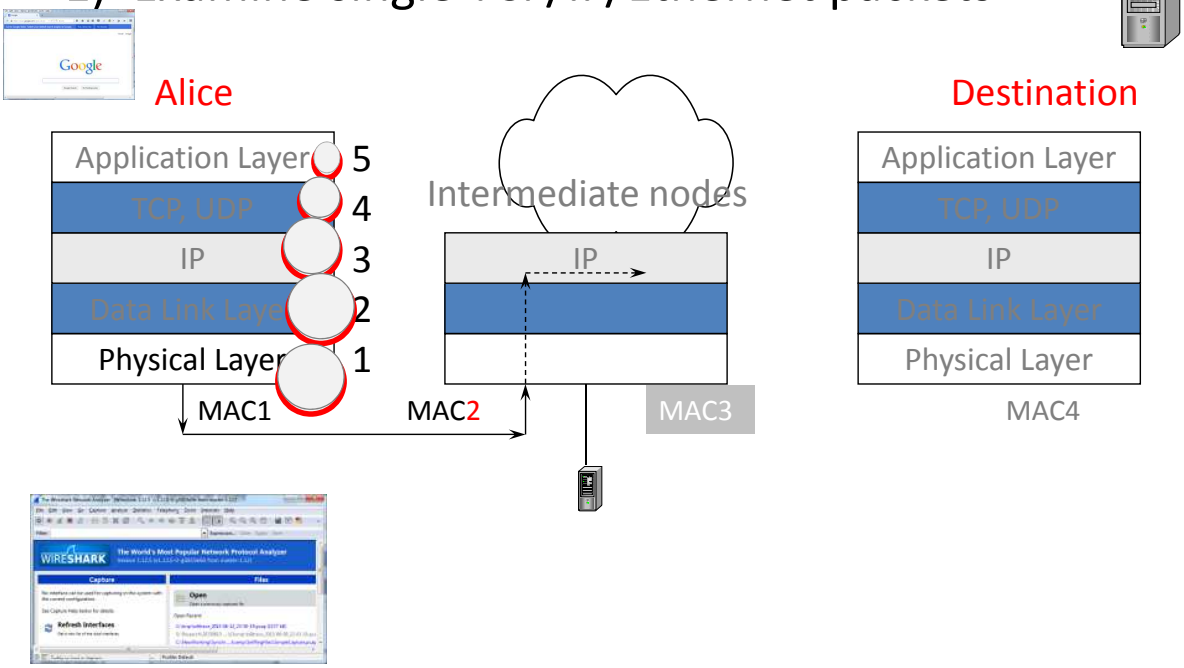
Prerequisite: TCP/IP Model (9/9)

- IP routing



Goals of this module (1/2)

1) Examine single TCP/IP/Ethernet packets



Goals of this module (2/2)

2) Find all packets related to one specific packet
3) Learn how to reduce packets for easy packet analysis
   - Statistics
     - Protocol hierarchy
     - HTTP requests
   - Conversations
   - Expressions
4) Learn how to find out attack packets (needle from a hay stack)

Two exercises

Exercise ❶: goals 1) ~ 2)

SimpleCapture.pcap, WebCapture.pcap

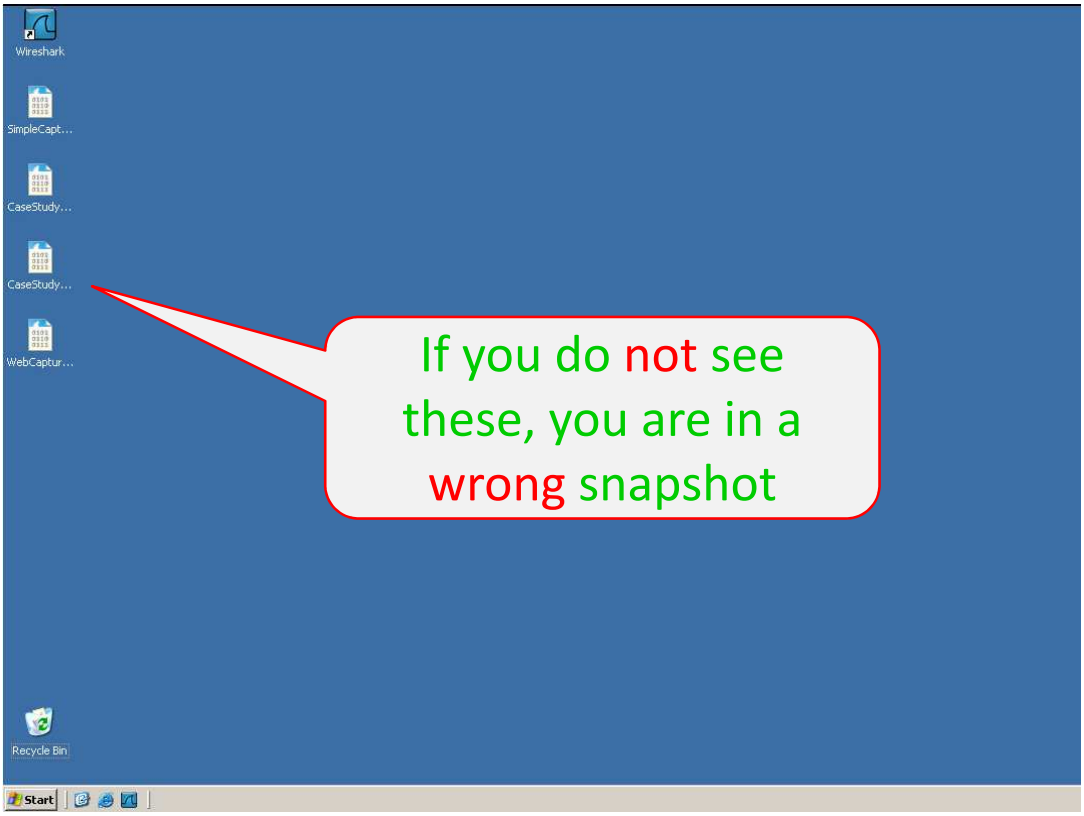
Exercise ❷: goals 3) ~ 4)

CaseStudy1.pcap, CaseStudy2.pcap

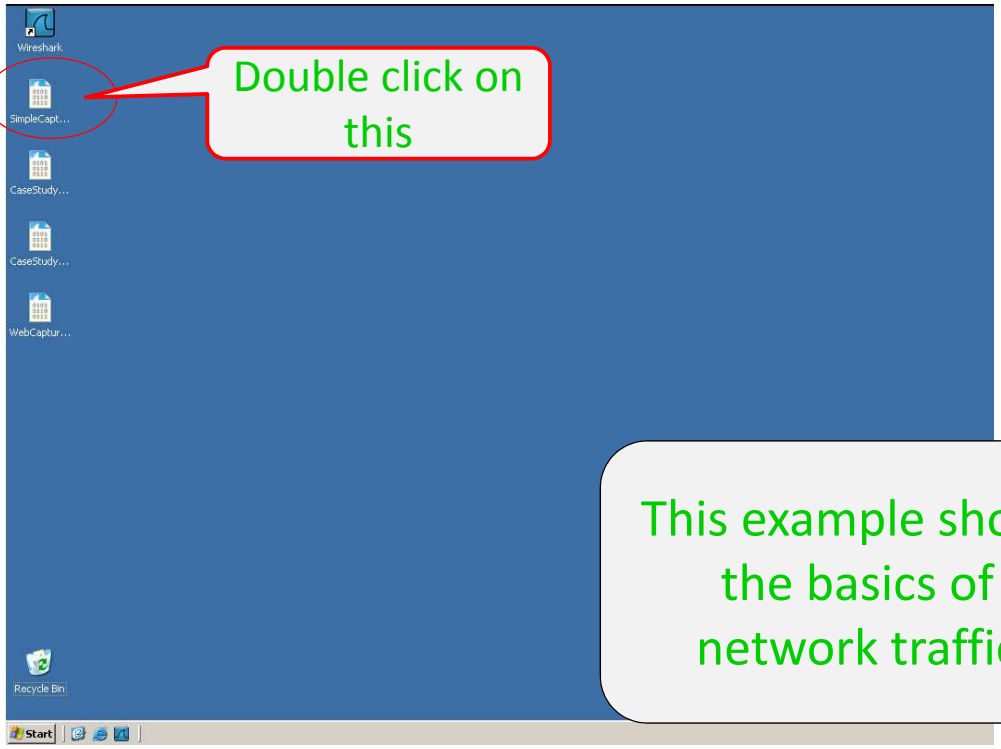
With what tool? Wireshark!

Step 0

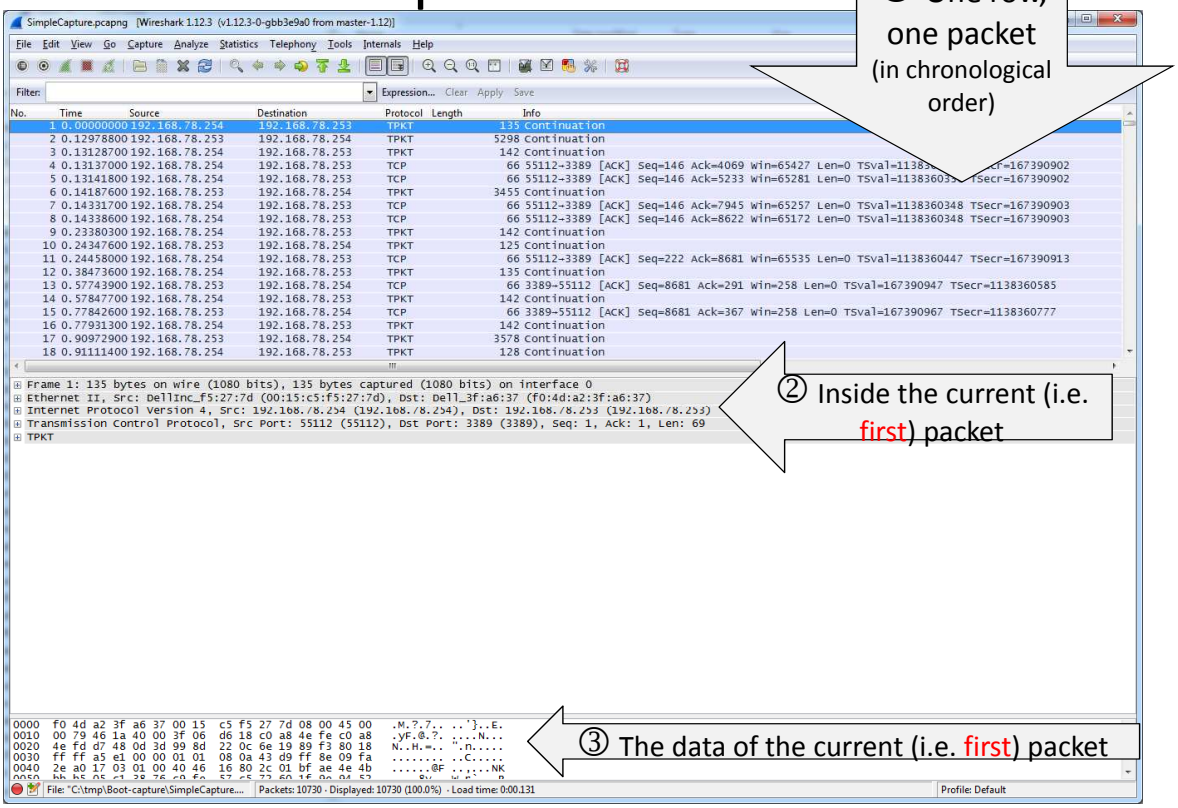
- Go to your VM
- Select the "Network Sniffing Exercise" snapshot
- Log in as administrator
- Password: password

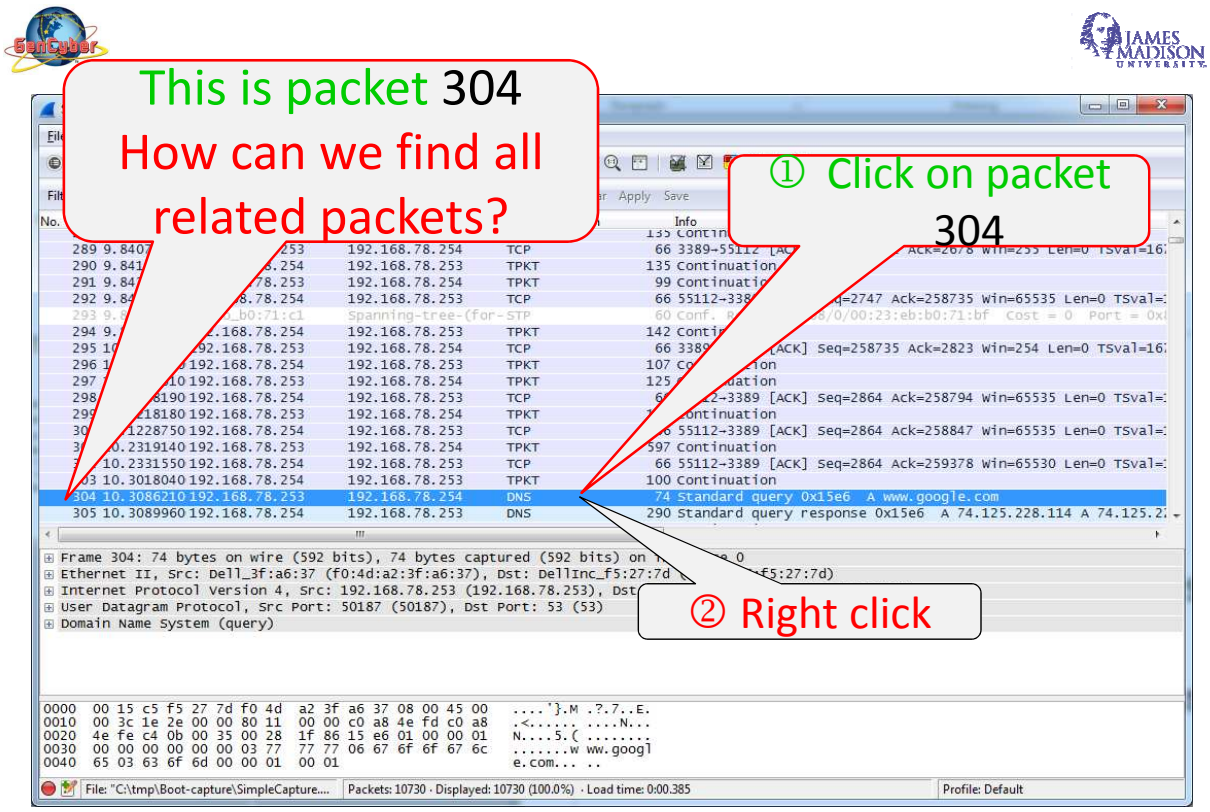


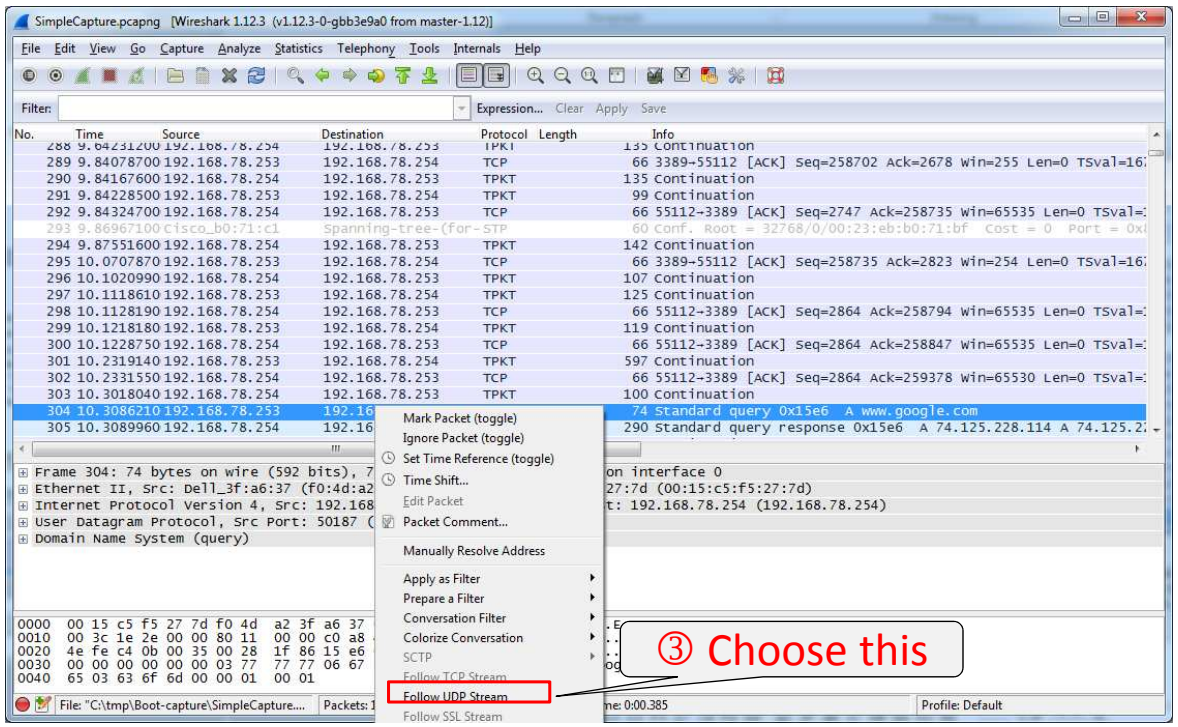
Exercise ❶: Basic Network Analysis

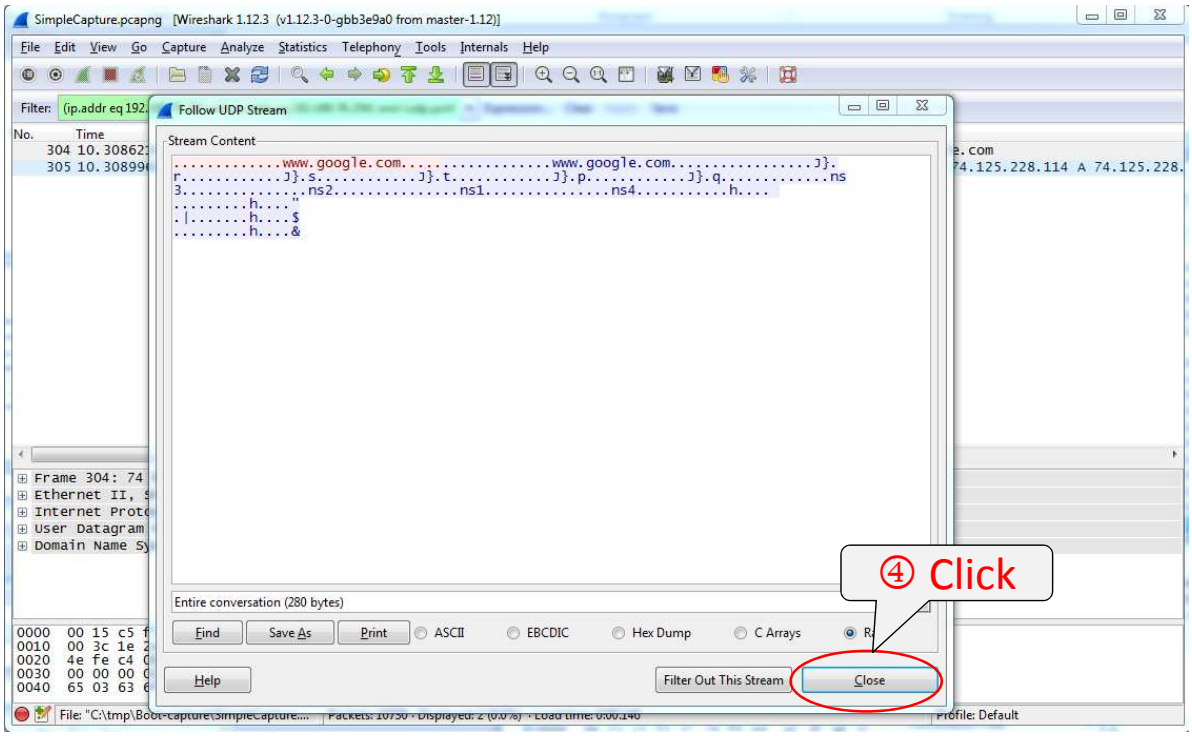


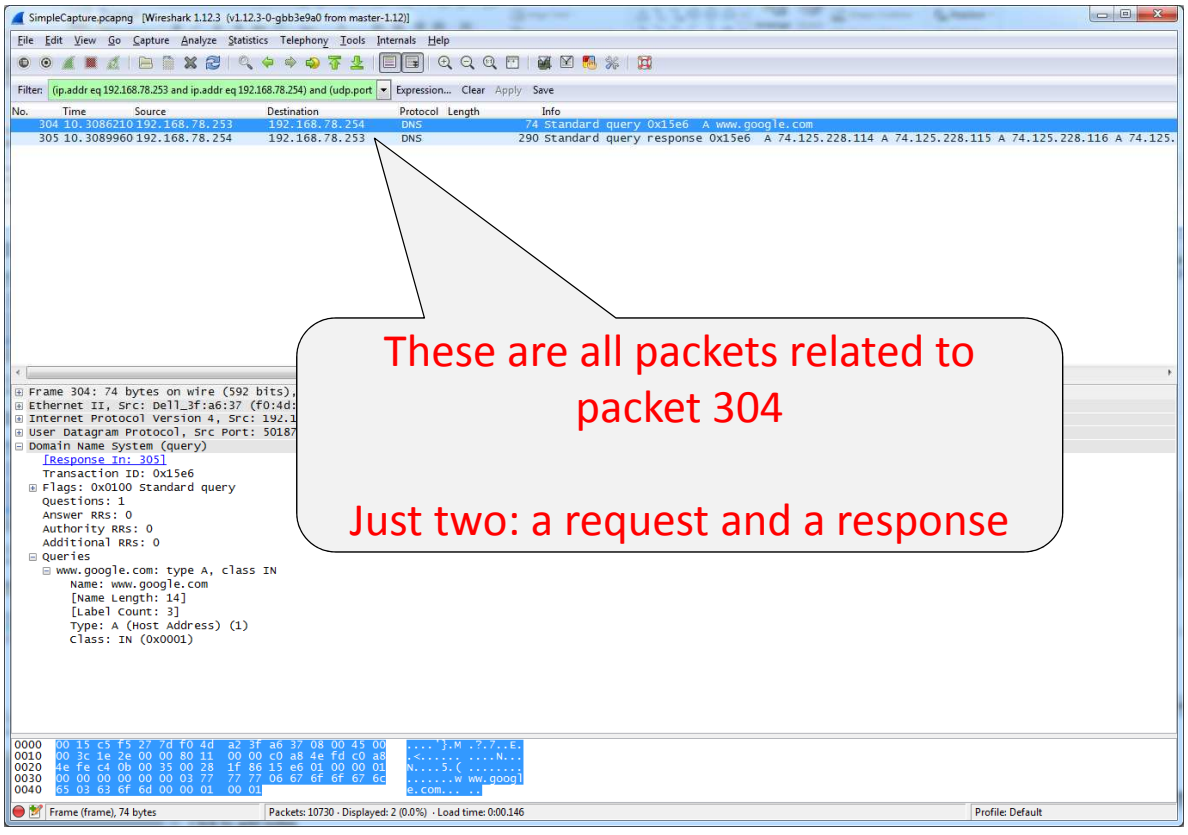
What does a packet look like?

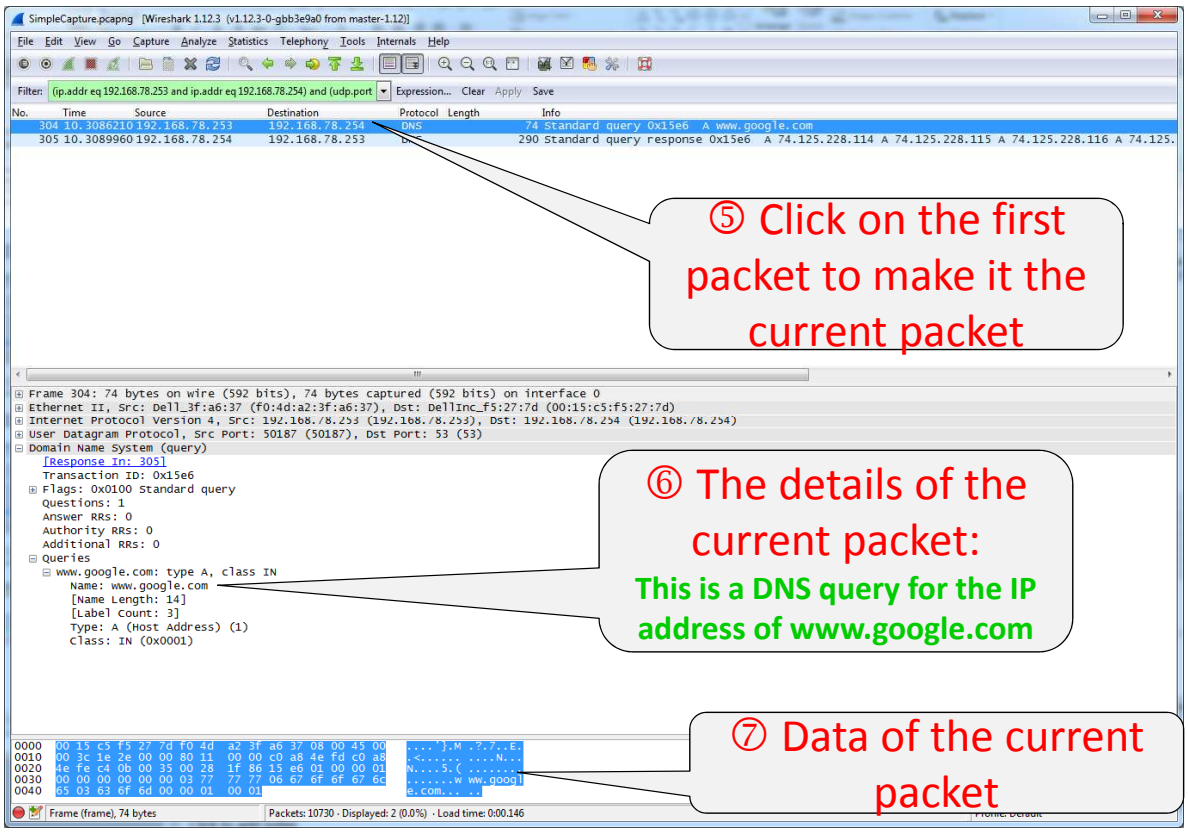


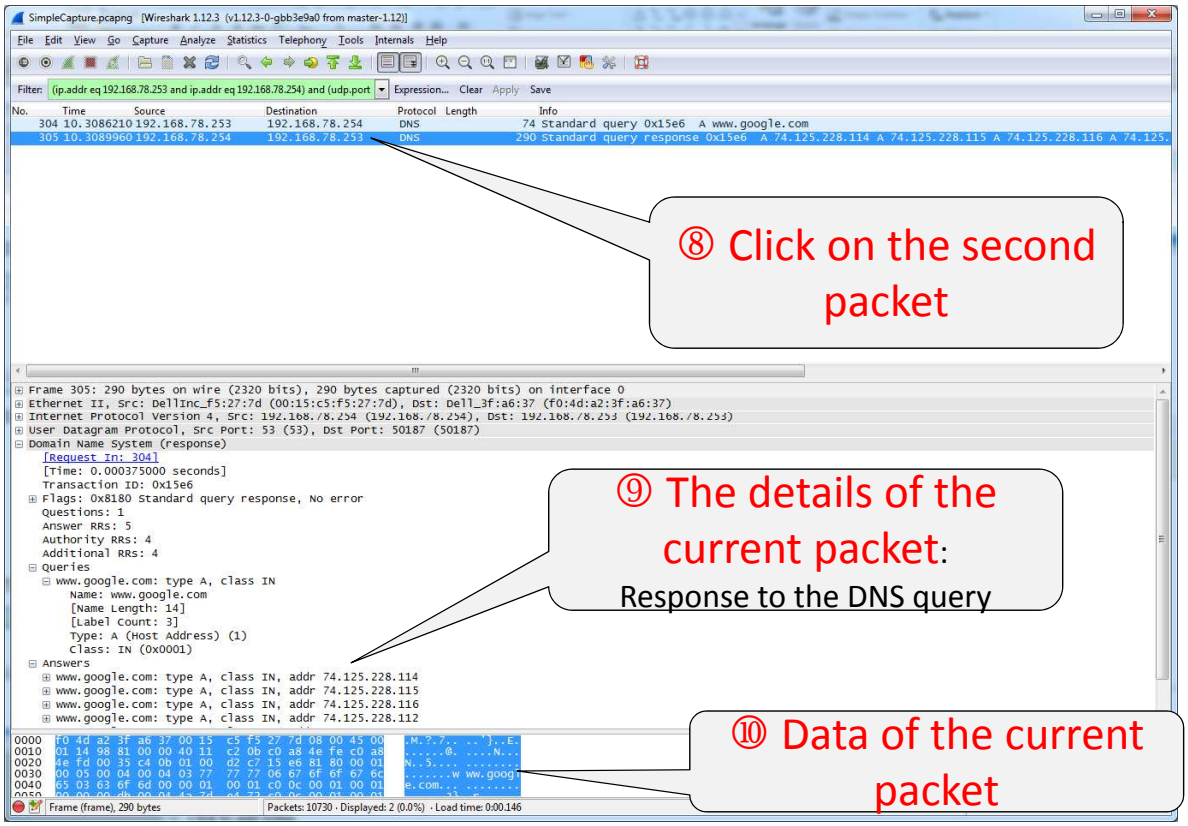

This is packet 304
How can we find all related packets?
① Click on packet 304
② Right click


③ Choose this

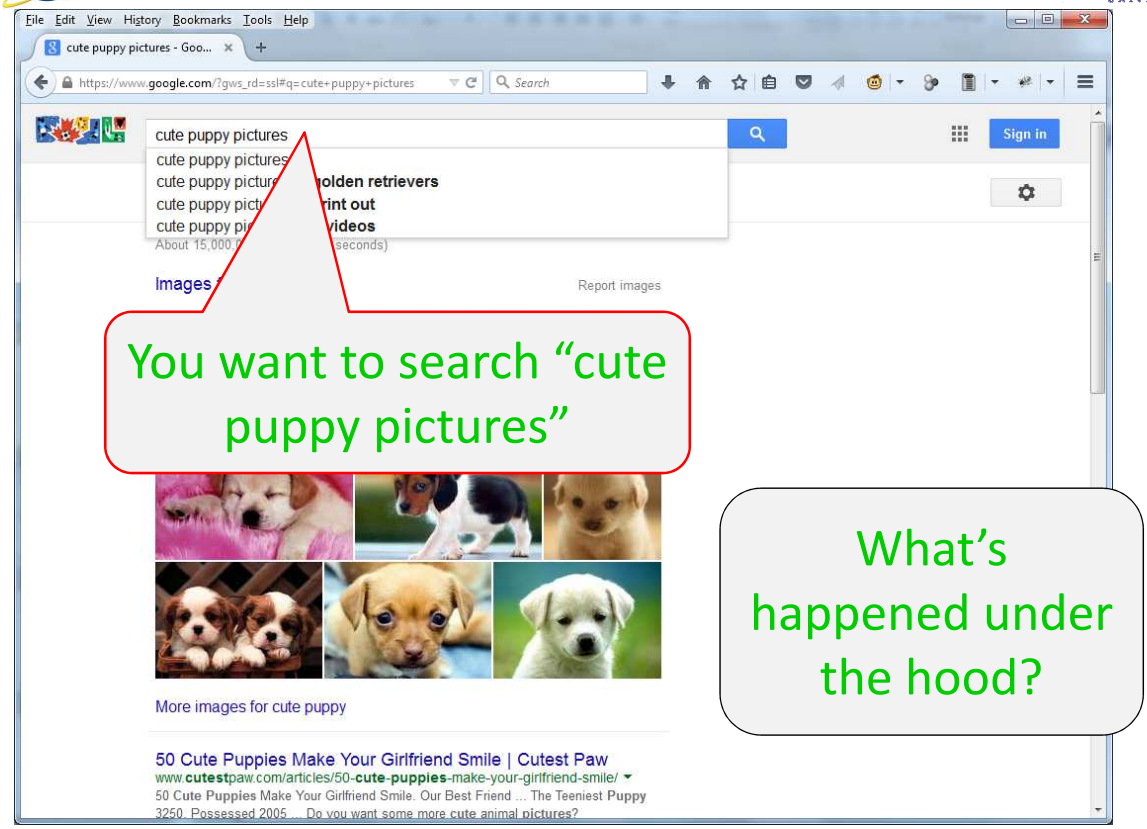

SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Follow UDP Stream
Stream Content
.....www.google.com.....www.google.com.....J}.
r.....J}.s.....J}.t.....J}.p.....J}.q.....ns
3.....ns2.....ns1.....ns4.....h....
.....h....
.|.....h....$
.....h....&
Entire conversation (280 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
Help
Filter Out This Stream
Close
④ Click

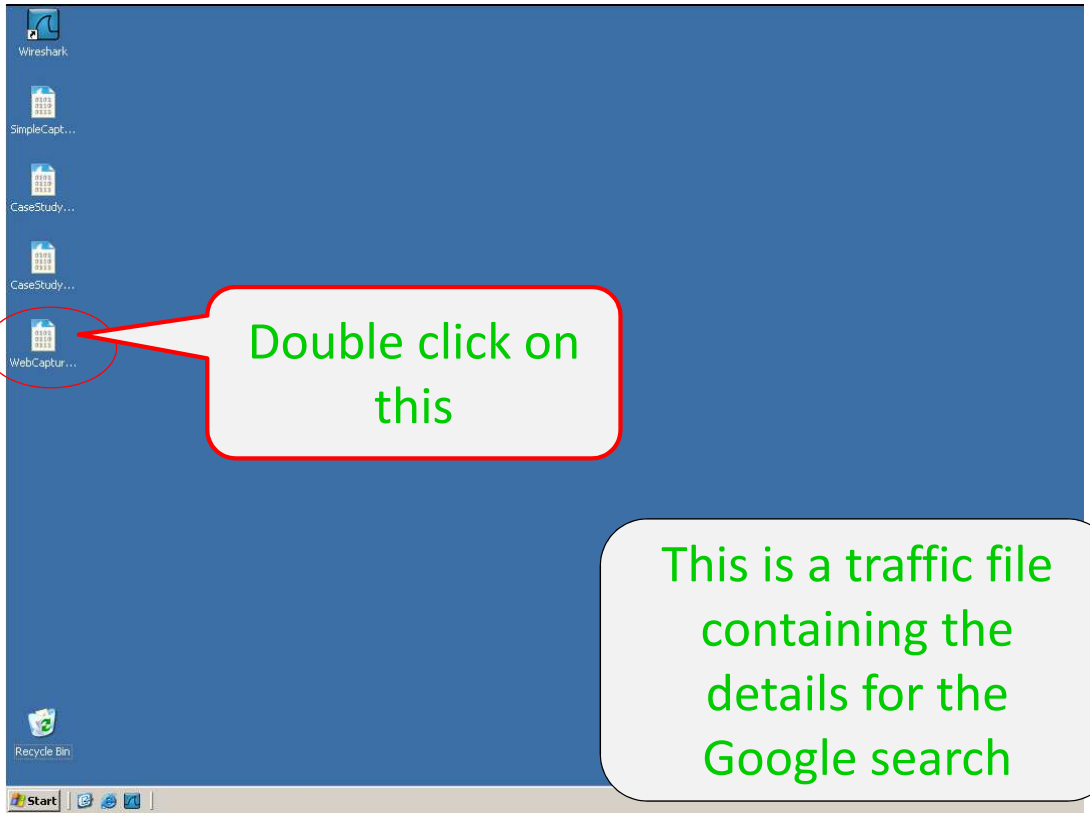

SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
304 10.3086210 192.168.78.253 192.168.78.254 DNS 74 Standard query 0x15e6 A www.google.com
305 10.3089960 192.168.78.254 192.168.78.253 DNS 290 Standard query response 0x15e6 A 74.125.228.114 A 74.125.228.115 A 74.125.228.116 A 74.125.
These are all packets related to packet 304
Just two: a request and a response

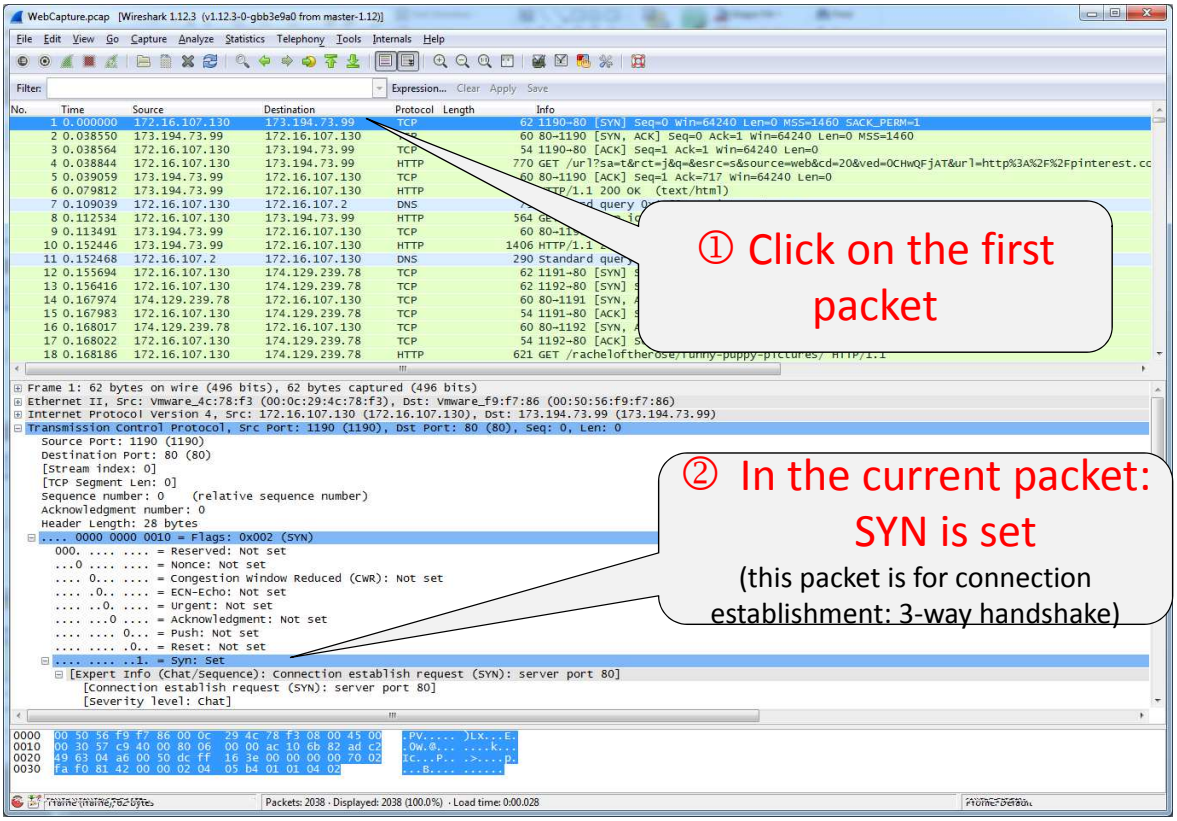

SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: (ip.addr eq 192.168.78.253 and ip.addr eq 192.168.78.254) and (udp.port
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
304 10.3086210 192.168.78.253 192.168.78.254 DNS 74 Standard query 0x15e6 A www.google.com
305 10.3089960 192.168.78.254 192.168.78.253 290 Standard query response 0x15e6 A 74.125.228.114 A 74.125.228.115 A 74.125.228.116 A 74.125.
⑤ Click on the first packet to make it the current packet
Frame 304: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: Dell_3f:a6:37 (f0:4d:a2:3f:a6:37), Dst: DellInc_f5:27:7d (00:15:c5:f5:27:7d)
Internet Protocol Version 4, Src: 192.168.78.253 (192.168.78.253), Dst: 192.168.78.254 (192.168.78.254)
User Datagram Protocol, Src Port: 50187 (50187), Dst Port: 53 (53)
Domain Name System (query)
[Response In: 305]
Transaction ID: 0x15e6
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.google.com: type A, class IN
Name: www.google.com
[Name Length: 14]
[Label Count: 3]
Type: A (Host Address) (1)
Class: IN (0x0001)
⑥ The details of the current packet:
This is a DNS query for the IP address of www.google.com
0000 00 15 c5 f5 27 7d f0 4d a2 3f a6 37 08 00 45 00 ....'}.M .?.7..E.
0010 00 3c 1e 2e 00 00 80 11 00 00 c0 a8 4e fd c0 a8 .<...... ....N...
0020 4e fe c4 0b 00 35 00 28 1f 86 15 e6 01 00 00 01 N....5.(........
0030 00 00 00 00 00 00 03 77 77 77 06 67 6f 6f 67 6c ....w ww.googl
0040 65 03 63 6f 6d 00 00 01 00 01 e.com... ..
⑦ Data of the current packet
Frame (frame), 74 bytes
Packets: 10730 · Displayed: 2 (0.0%) · Load time: 0:00.146

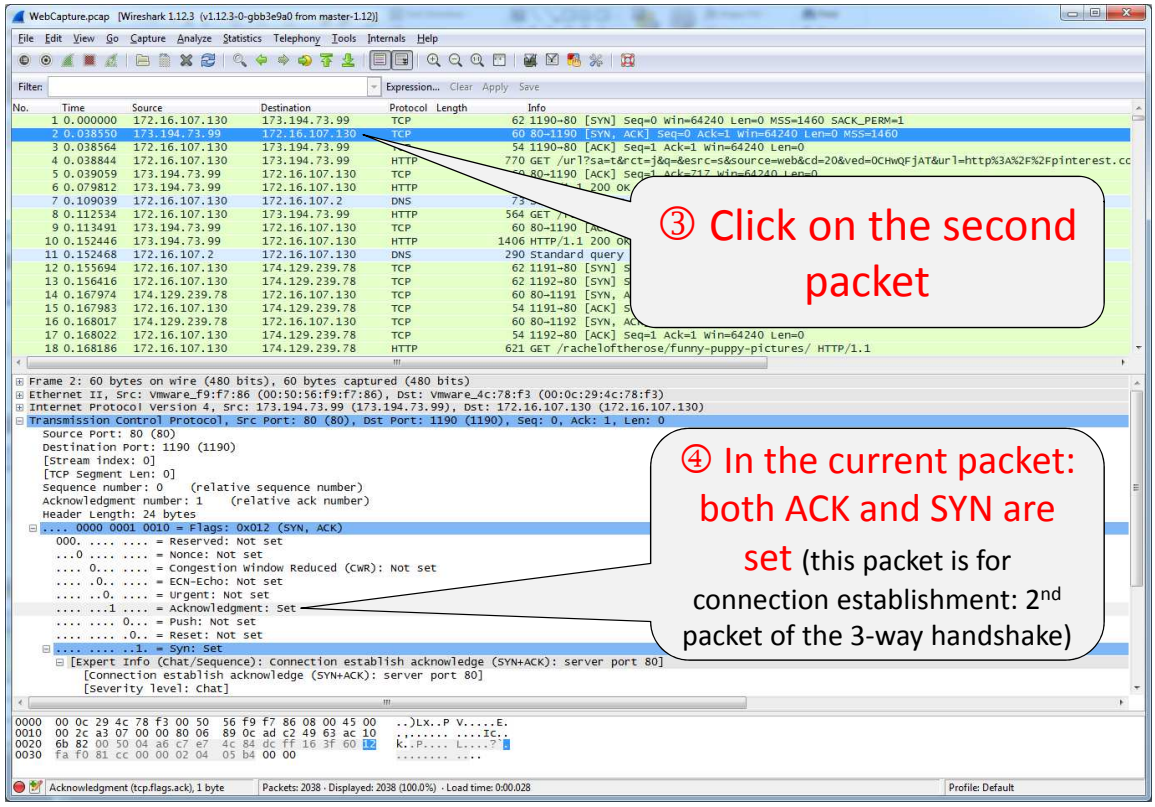

SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: (ip.addr eq 192.168.78.253 and ip.addr eq 192.168.78.254) and (udp.port
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
304 10.3086210 192.168.78.253 192.168.78.254 DNS 74 Standard query 0x15e6 A www.google.com
305 10.3089960 192.168.78.254 192.168.78.253 DNS 290 Standard query response 0x15e6 A 74.125.228.114 A 74.125.228.115 A 74.125.228.116 A 74.125.
⑧ Click on the second packet
Frame 305: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface 0
Ethernet II, Src: DellInc_f5:27:7d (00:15:c5:f5:27:7d), Dst: Dell_3f:a6:37 (f0:4d:a2:3f:a6:37)
Internet Protocol Version 4, Src: 192.168.78.254 (192.168.78.254), Dst: 192.168.78.253 (192.168.78.253)
User Datagram Protocol, Src Port: 53 (53), Dst Port: 50187 (50187)
Domain Name System (response)
[Request In: 304]
[Time: 0.000375000 seconds]
Transaction ID: 0x15e6
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 5
Authority RRs: 4
Additional RRs: 4
Queries
www.google.com: type A, class IN
Name: www.google.com
[Name Length: 14]
[Label Count: 3]
Type: A (Host Address) (1)
Class: IN (0x0001)
Answers
www.google.com: type A, class IN, addr 74.125.228.114
www.google.com: type A, class IN, addr 74.125.228.115
www.google.com: type A, class IN, addr 74.125.228.116
www.google.com: type A, class IN, addr 74.125.228.112
⑨ The details of the current packet:
Response to the DNS query
0000 f0 4d a2 3f a6 37 00 15 c5 f5 27 7d 08 00 45 00 .M.?.7.. ..'}..E.
0010 01 14 98 81 00 00 40 11 c2 0b c0 a8 4e fe c0 a8 ......@. ....N...
0020 4e fd 00 35 c4 0b 01 00 d2 c7 15 e6 81 80 00 01 N..5.... ....
0030 00 05 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c ....w ww.googl
0040 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 01 e.com... ....
⑩ Data of the current packet
Frame (frame), 290 bytes
Packets: 10730 · Displayed: 2 (0.0%) · Load time: 0:00.146

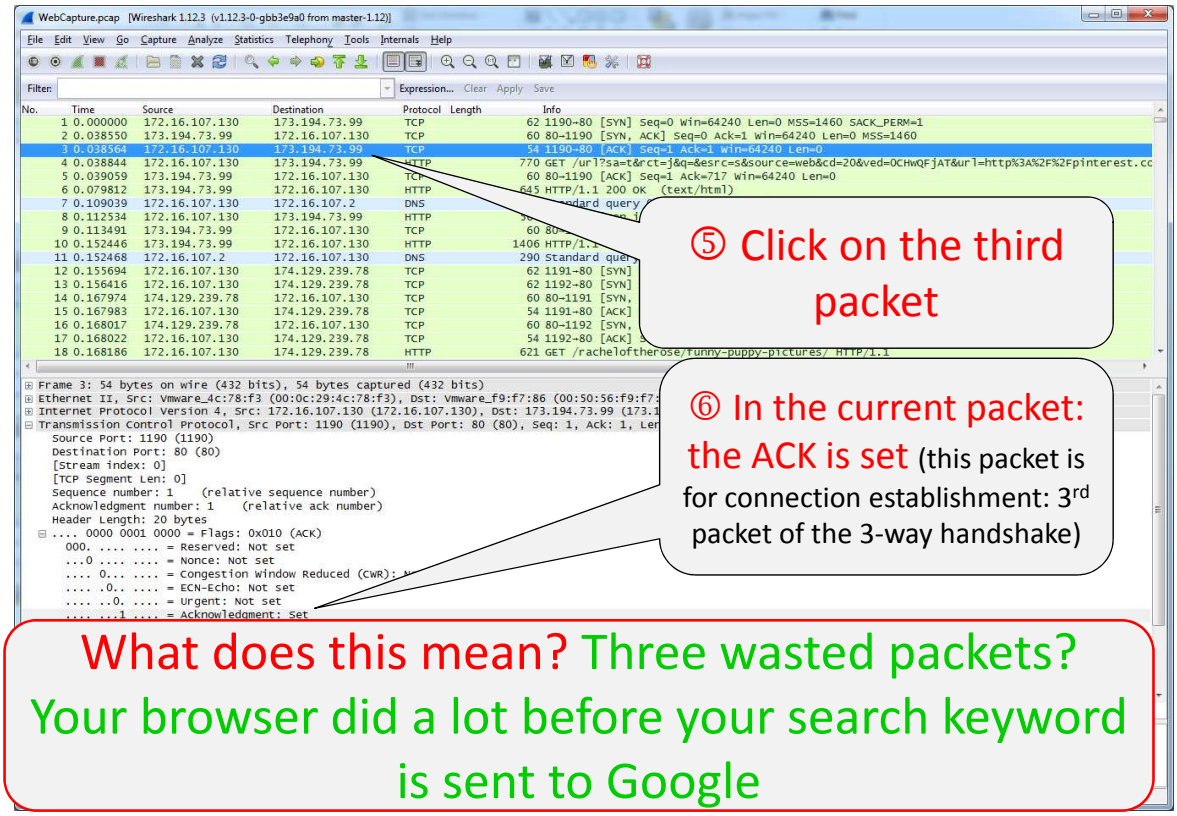
So far, so good?




Wireshark
SimpleCapt...
CaseStudy...
CaseStudy...
WebCaptur...
Recycle Bin
Start
Double click on this
This is a traffic file containing the details for the Google search

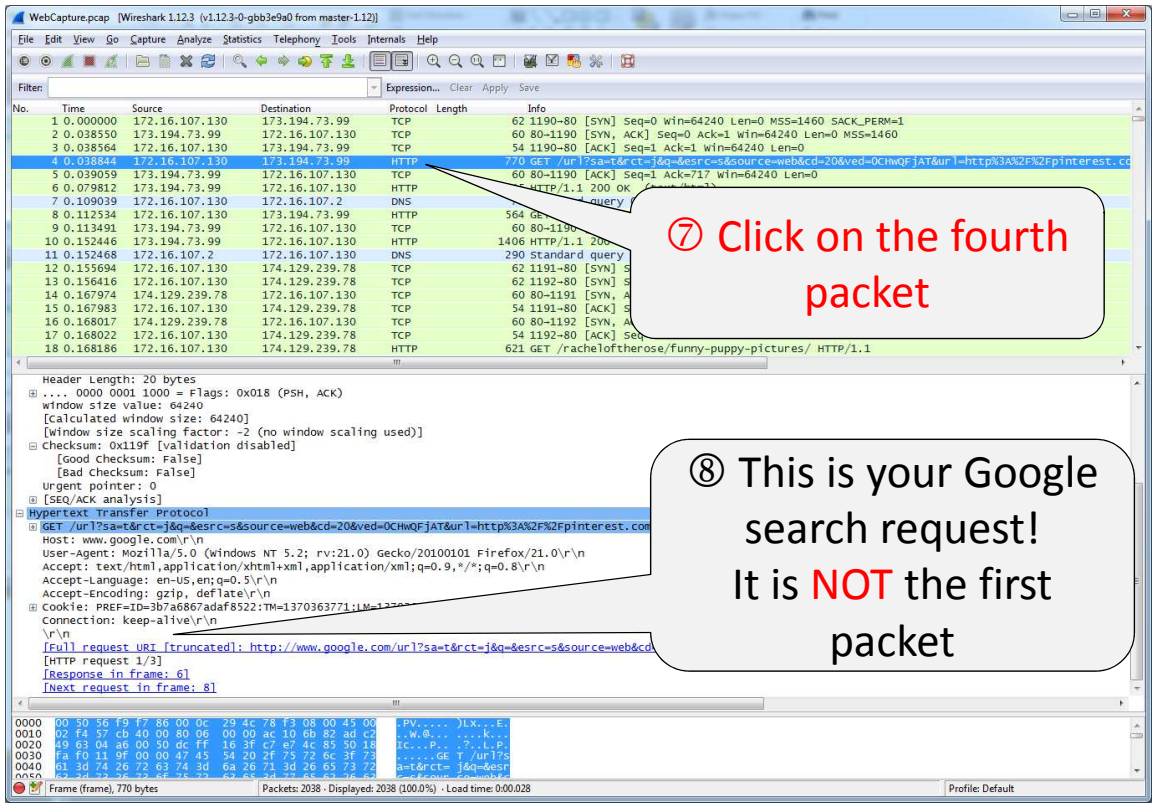

WebCapture.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000 172.16.107.130 173.194.73.99 TCP 62 1190→80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
2 0.038550 173.194.73.99 172.16.107.130 60 80→1190 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3 0.038564 172.16.107.130 173.194.73.99 TCP 54 1190→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4 0.038844 172.16.107.130 173.194.73.99 HTTP 770 GET /url?sa=t&rct=j&q=&esrc=s&source=web&cd=20&ved=0CHwQFjAT&url=http%3A%2F%2Fpinterest.cc
5 0.039059 173.194.73.99 172.16.107.130 TCP 60 80→1190 [ACK] Seq=1 Ack=717 Win=64240 Len=0
6 0.079812 173.194.73.99 172.16.107.130 HTTP TTP/1.1 200 OK (text/html)
7 0.109039 172.16.107.130 172.16.107.2 DNS d query 0
8 0.112534 172.16.107.130 173.194.73.99 HTTP 564 GE
9 0.113491 173.194.73.99 172.16.107.130 TCP 60 80→11
10 0.152446 173.194.73.99 172.16.107.130 HTTP 1406 HTTP/1.1
11 0.152468 172.16.107.2 172.16.107.130 DNS 290 Standard que
12 0.155694 172.16.107.130 174.129.239.78 TCP 62 1191→80 [SYN]
13 0.156416 172.16.107.130 174.129.239.78 TCP 62 1192→80 [SYN]
14 0.167974 174.129.239.78 172.16.107.130 TCP 60 80→1191 [SYN,
15 0.167983 172.16.107.130 174.129.239.78 TCP 54 1191→80 [ACK]
16 0.168017 174.129.239.78 172.16.107.130 TCP 60 80→1192 [SYN,
17 0.168022 172.16.107.130 174.129.239.78 TCP 54 1192→80 [ACK]
18 0.168186 172.16.107.130 174.129.239.78 HTTP 621 GET /racheloftherose/funny-puppy-pictures/ HTTP/1.1
Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: Vmware_4c:78:f3 (00:0c:29:4c:78:f3), Dst: Vmware_f9:f7:86 (00:50:56:f9:f7:86)
Internet Protocol Version 4, Src: 172.16.107.130 (172.16.107.130), Dst: 173.194.73.99 (173.194.73.99)
Transmission Control Protocol, Src Port: 1190 (1190), Dst Port: 80 (80), Seq: 0, Len: 0
Source Port: 1190 (1190)
Destination Port: 80 (80)
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 0
Header Length: 28 bytes
.... 0000 0000 0010 = Flags: 0x002 (SYN)
000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
....0 .... = Acknowledgment: Not set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
....1. = Syn: Set
[Expert Info (Chat/Sequence): Connection establish request (SYN): server port 80]
[Connection establish request (SYN): server port 80]
[Severity level: Chat]
0000 00 50 56 f9 f7 86 00 0c 29 4c 78 f3 08 00 45 00 .PV.....)Lx...E.
0010 00 30 57 c9 40 00 80 06 00 00 ac 10 6b 82 ad c2 .0W.@... ....k...
0020 49 63 04 a6 00 50 dc ff 16 3e 00 00 00 00 70 02 Ic...P.. .>....p.
0030 fa f0 81 42 00 00 02 04 05 b4 01 01 04 02 ...B.... ......
Packets: 2038 · Displayed: 2038 (100.0%) · Load time: 0:00.028
① Click on the first packet
② In the current packet: SYN is set
(this packet is for connection establishment: 3-way handshake)

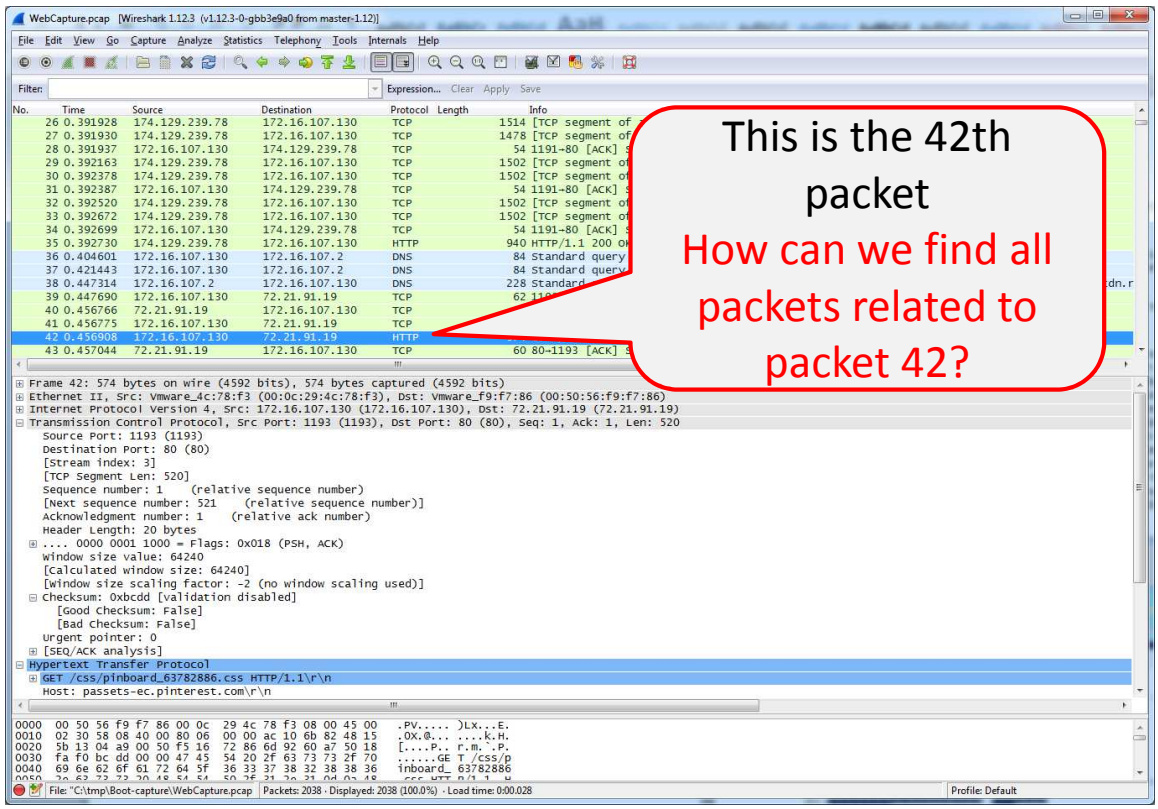


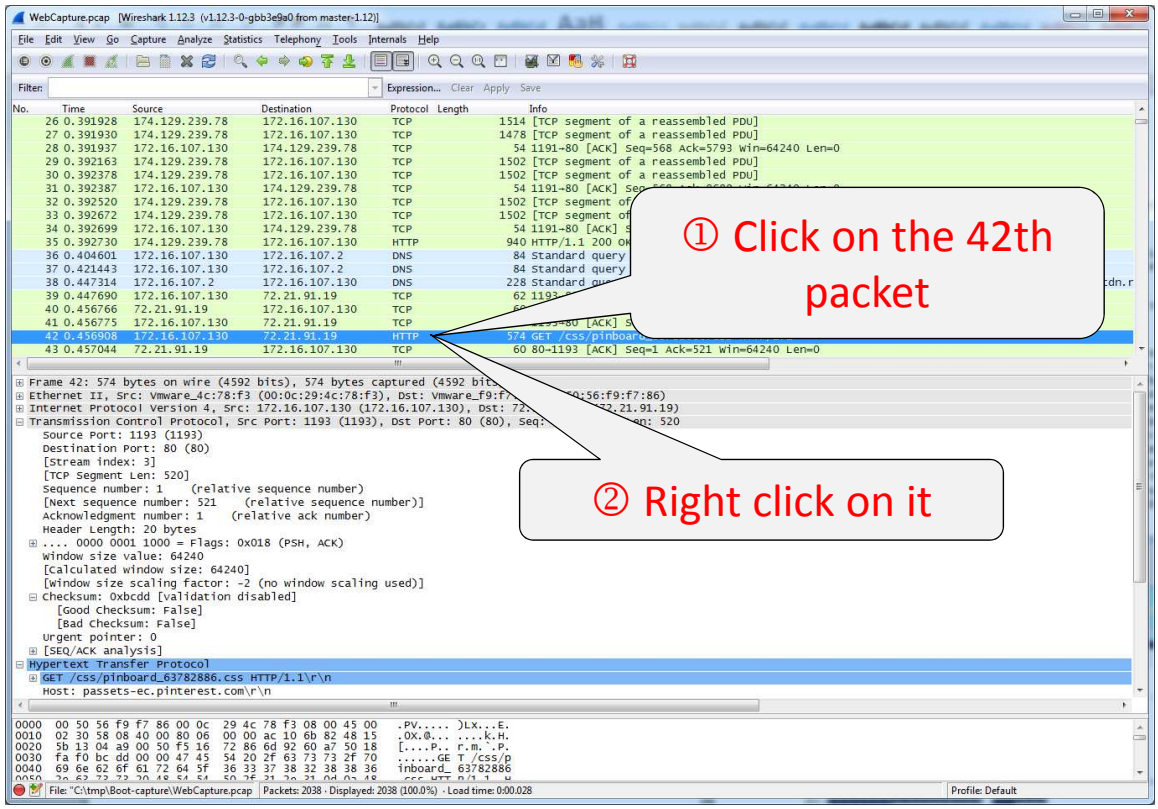


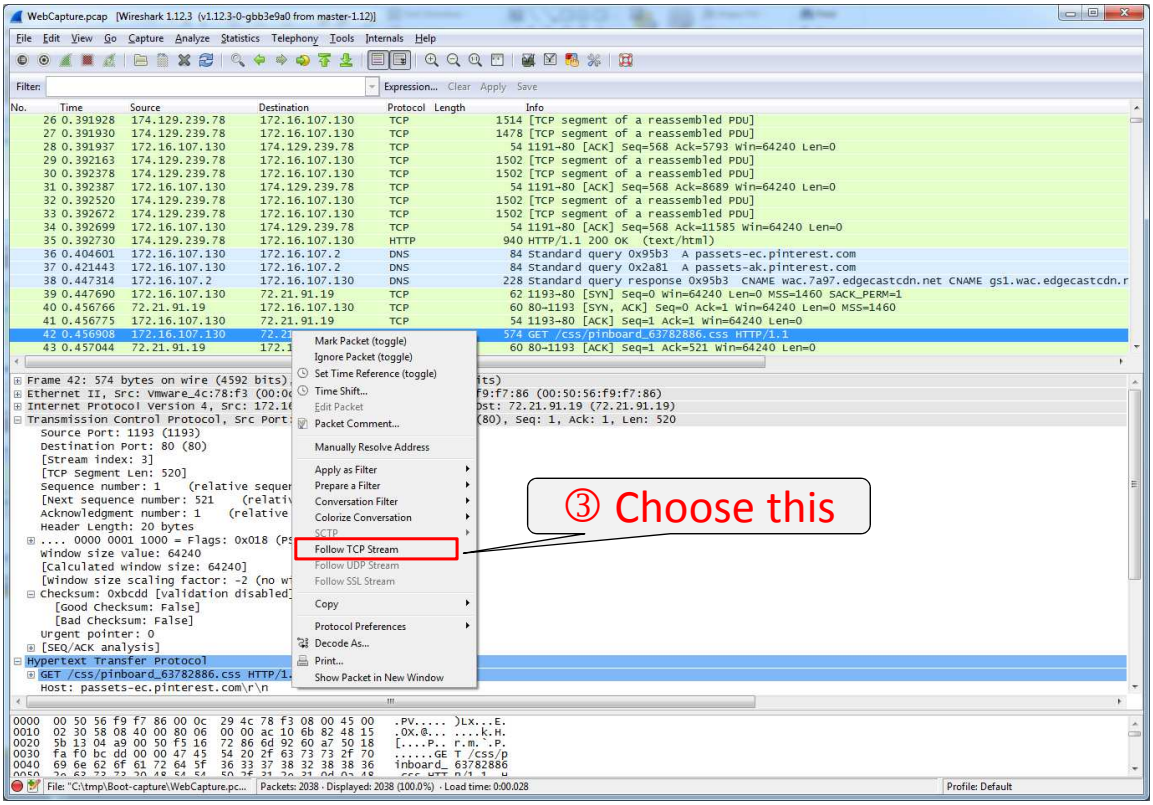
What does this mean? Three wasted packets?

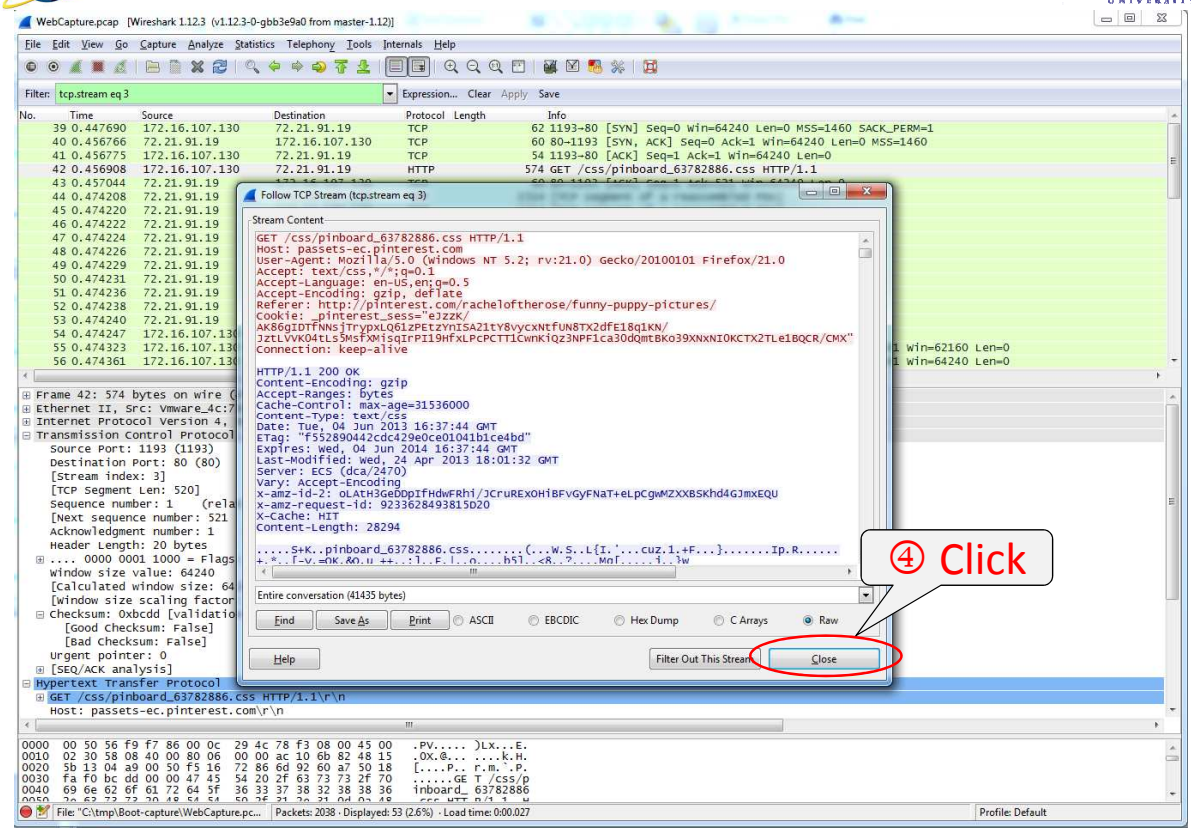
Your browser did a lot before your search keyword is sent to Google

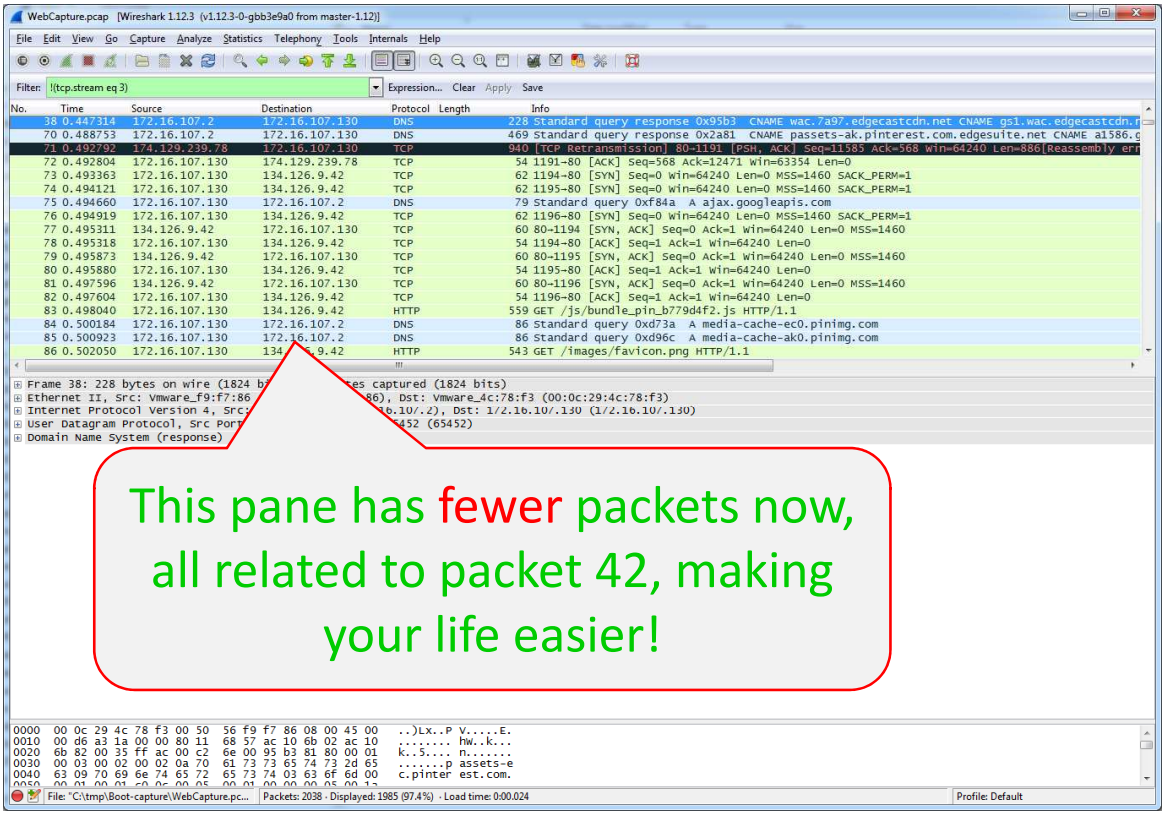

⑦ Click on the fourth packet
⑧ This is your Google search request!
It is NOT the first packet


This is the 42th packet
How can we find all packets related to packet 42?


① Click on the 42th packet
② Right click on it


③ Choose this
Mark Packet (toggle)
Ignore Packet (toggle)
Set Time Reference (toggle)
Time Shift...
Edit Packet
Packet Comment...
Manually Resolve Address
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow TCP Stream
Follow UDP Stream
Follow SSL Stream
Copy
Protocol Preferences
Decode As...
Print...
Show Packet in New Window


④ Click

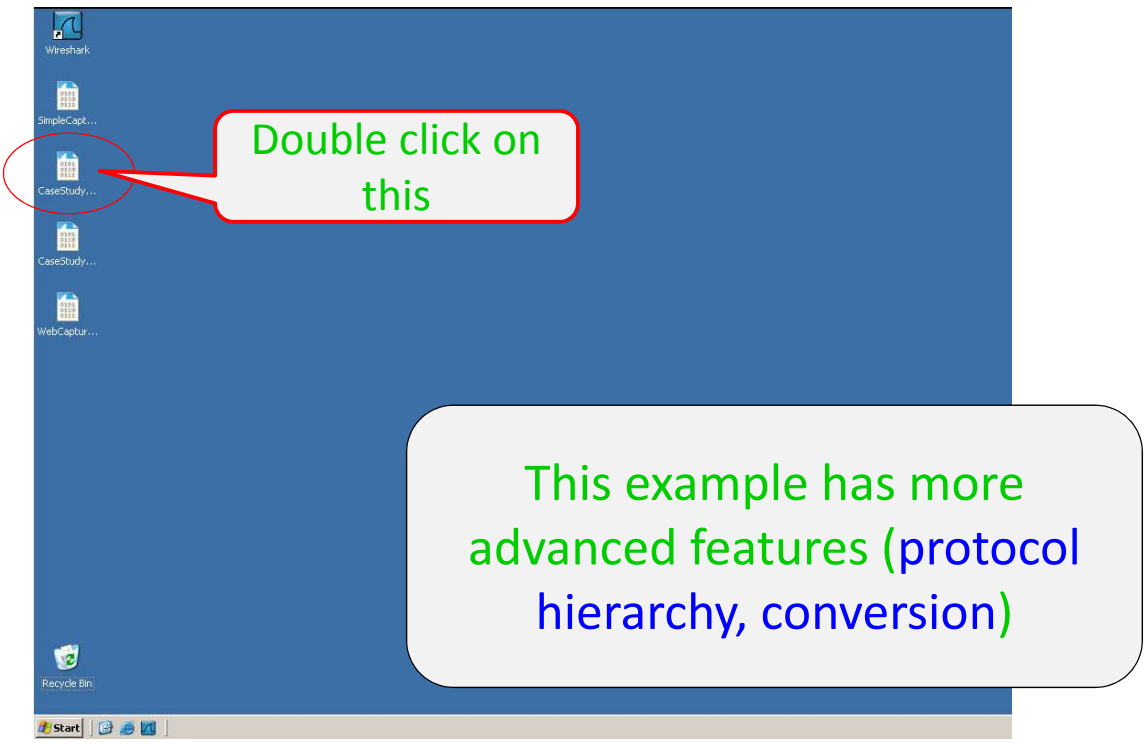

This pane has fewer packets now, all related to packet 42, making your life easier!

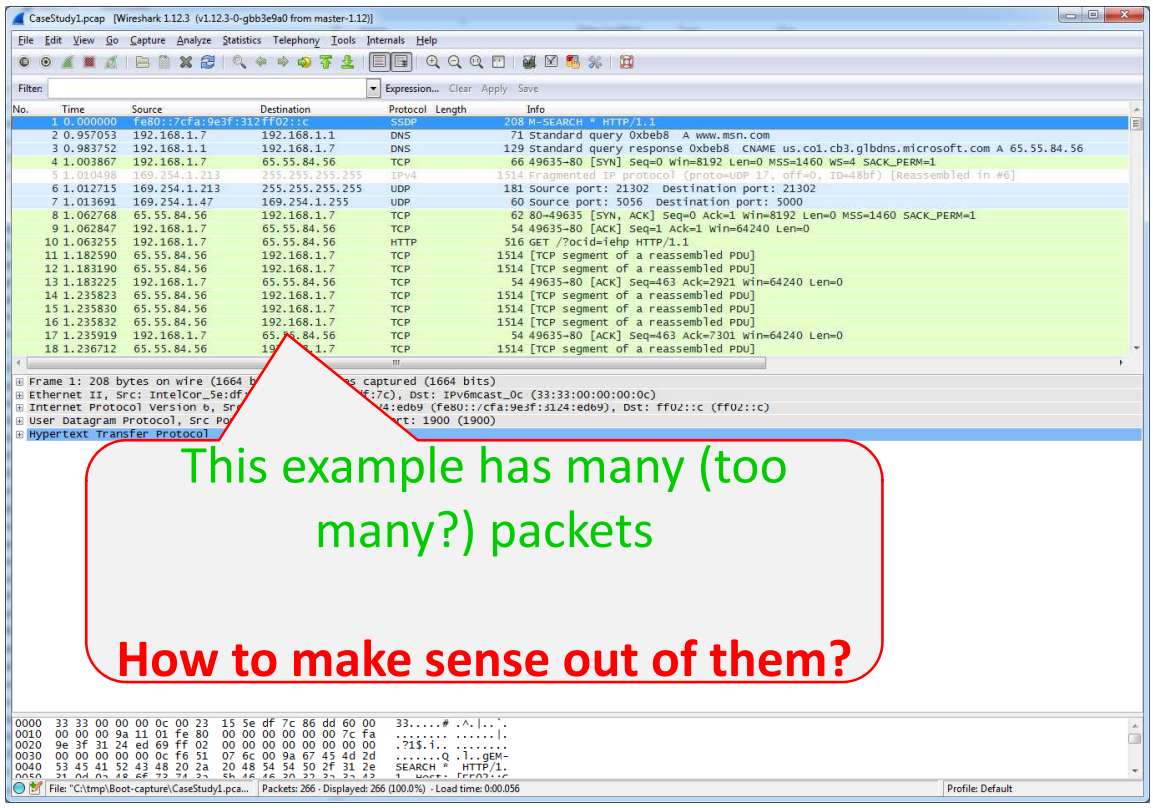
Quiz

Everybody likes a quiz!

- Wireshark is a popular tool for:
  a) Testing web applications for vulnerabilities
  b) Cracking WEP encryption used in older wireless networks
  c) Analyzing the contents of network traffic
  d) Crafting phishing e-mails
  e) None of the above

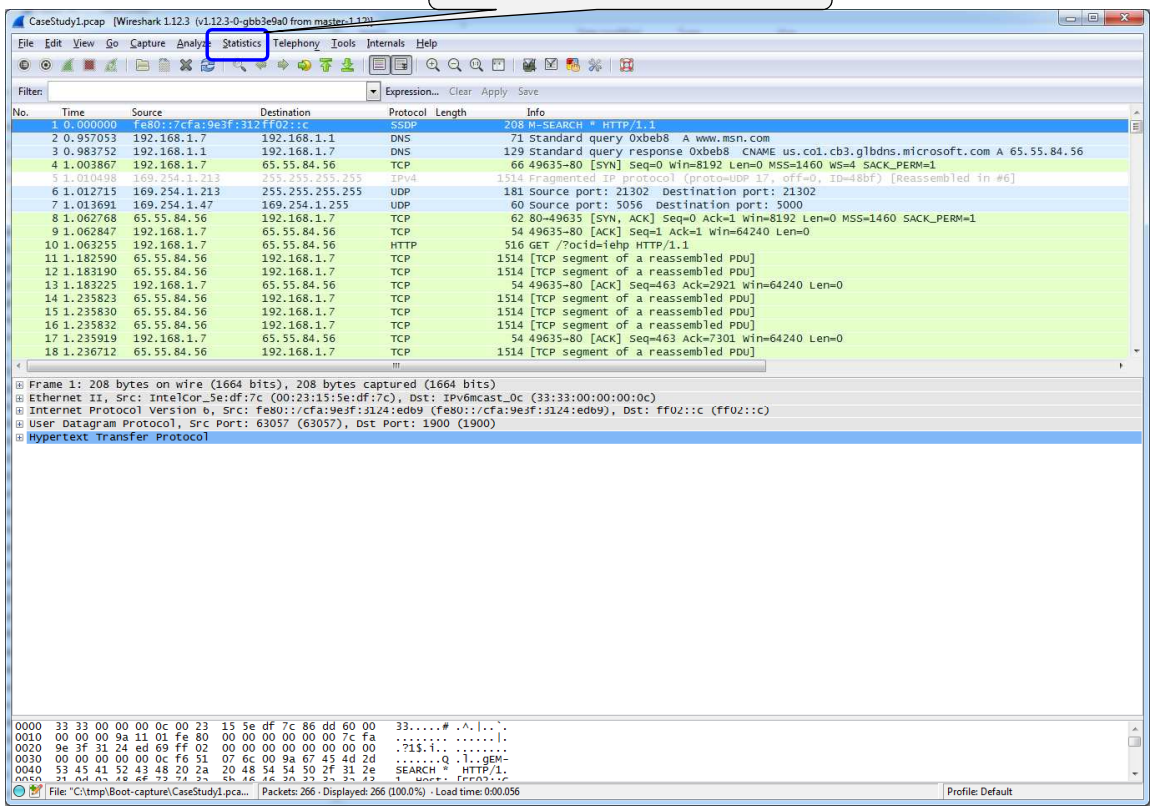
Exercise ❷

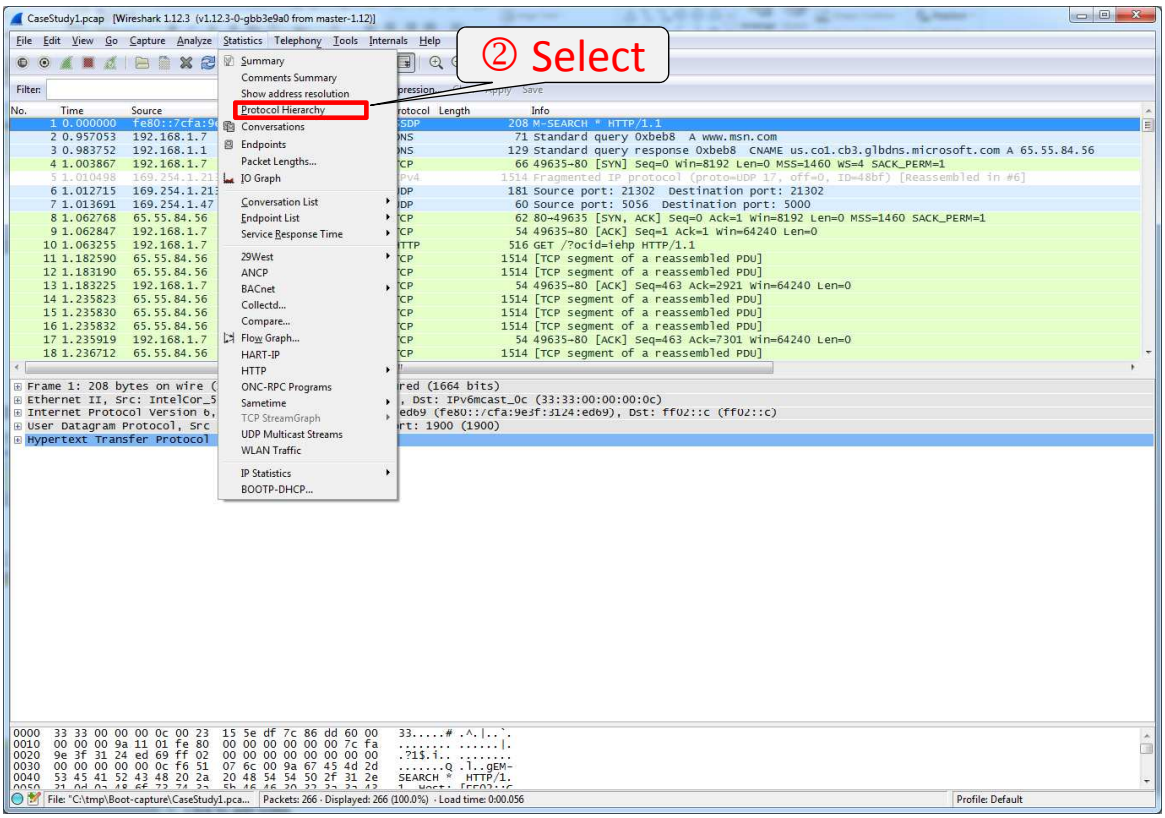


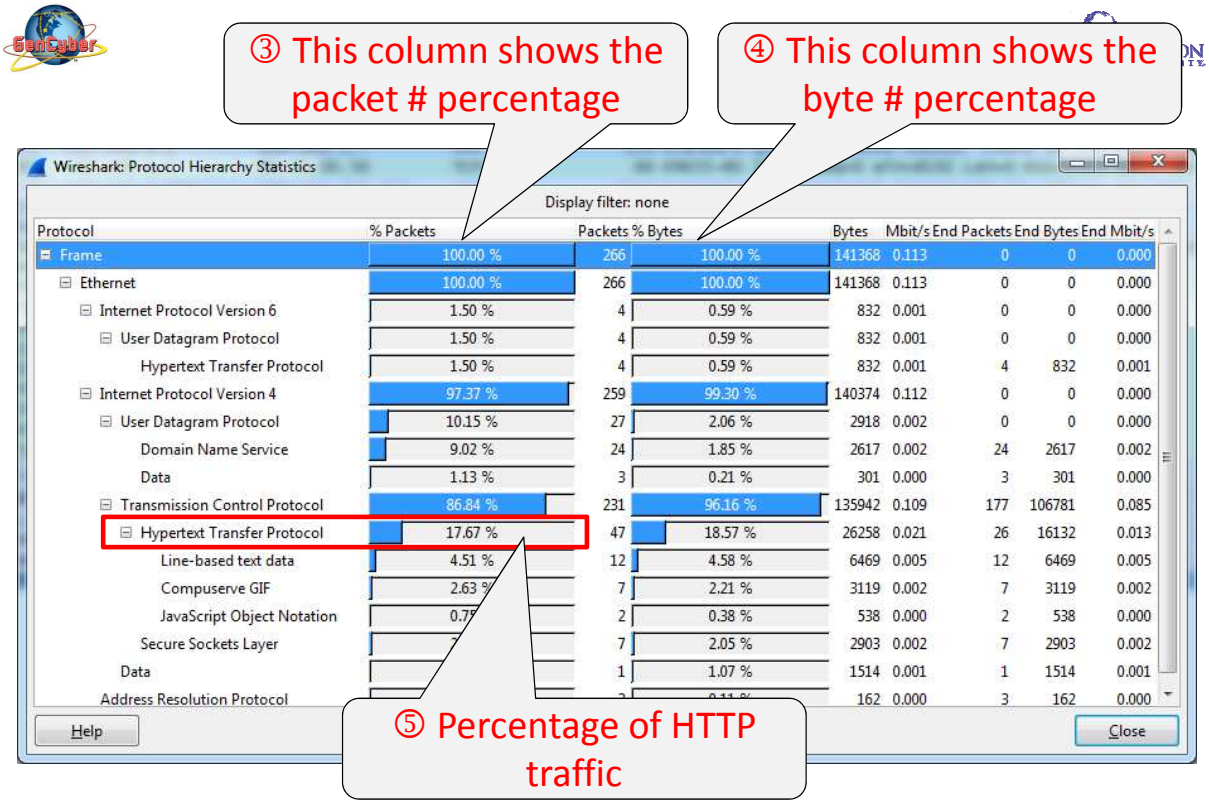


This example has many (too many?) packets

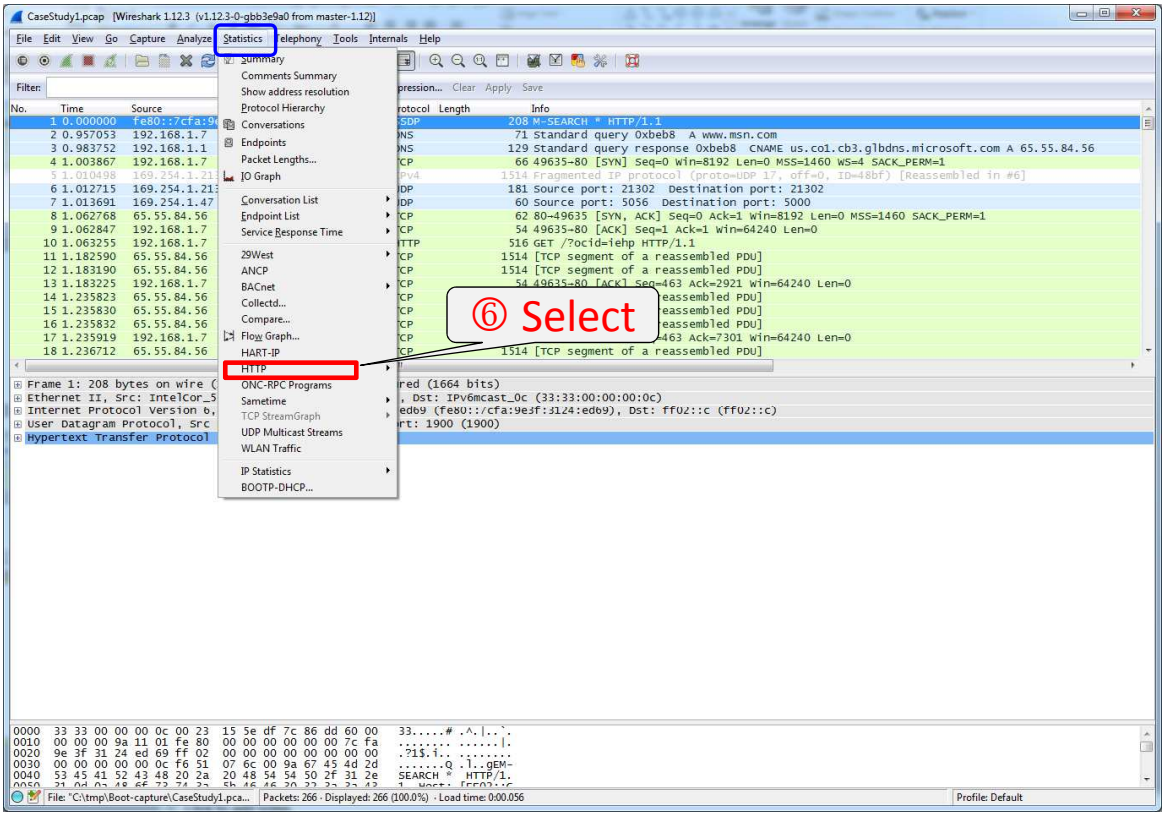
How to make sense out of them?

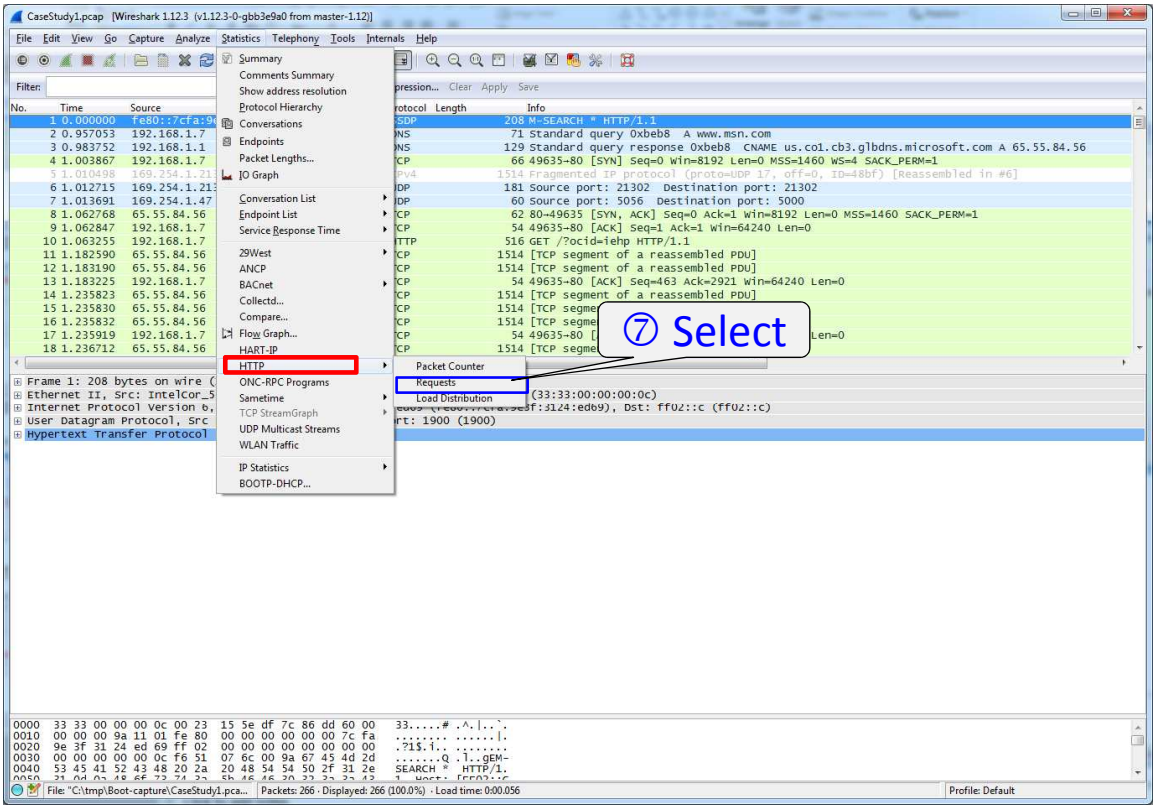


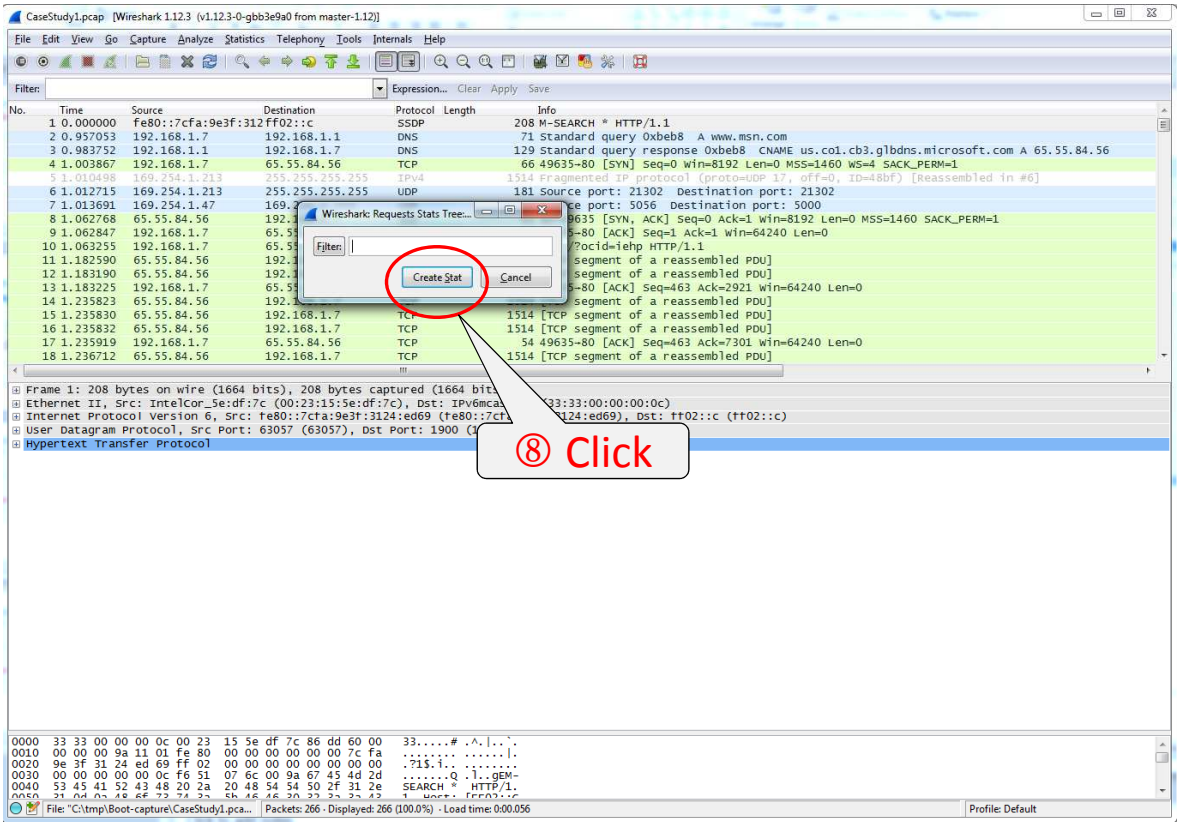


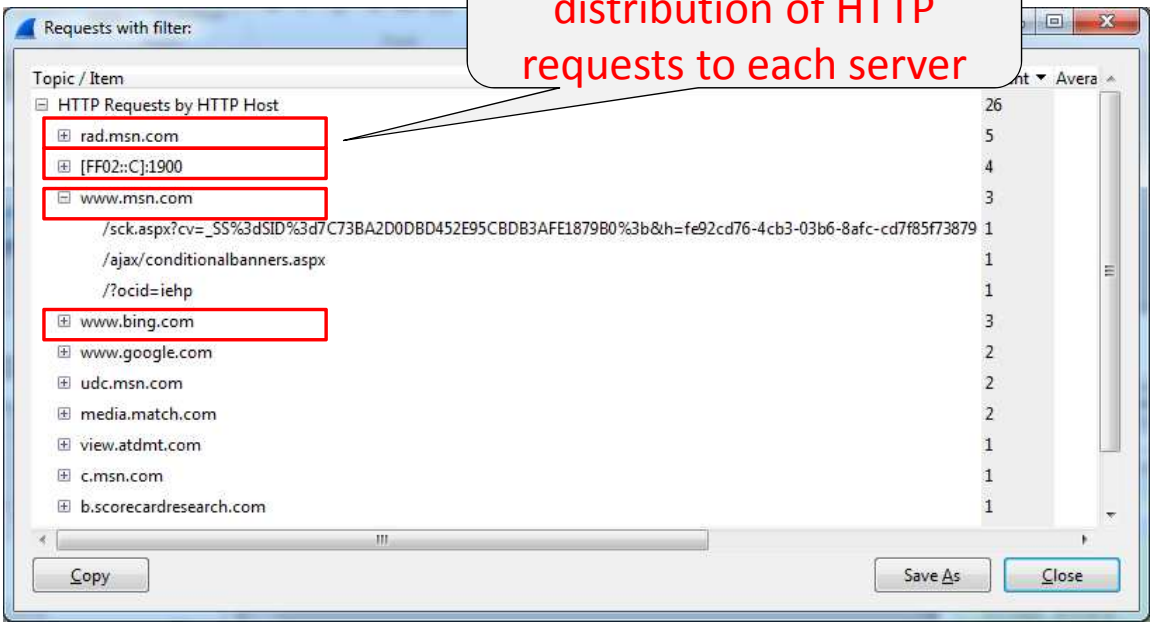


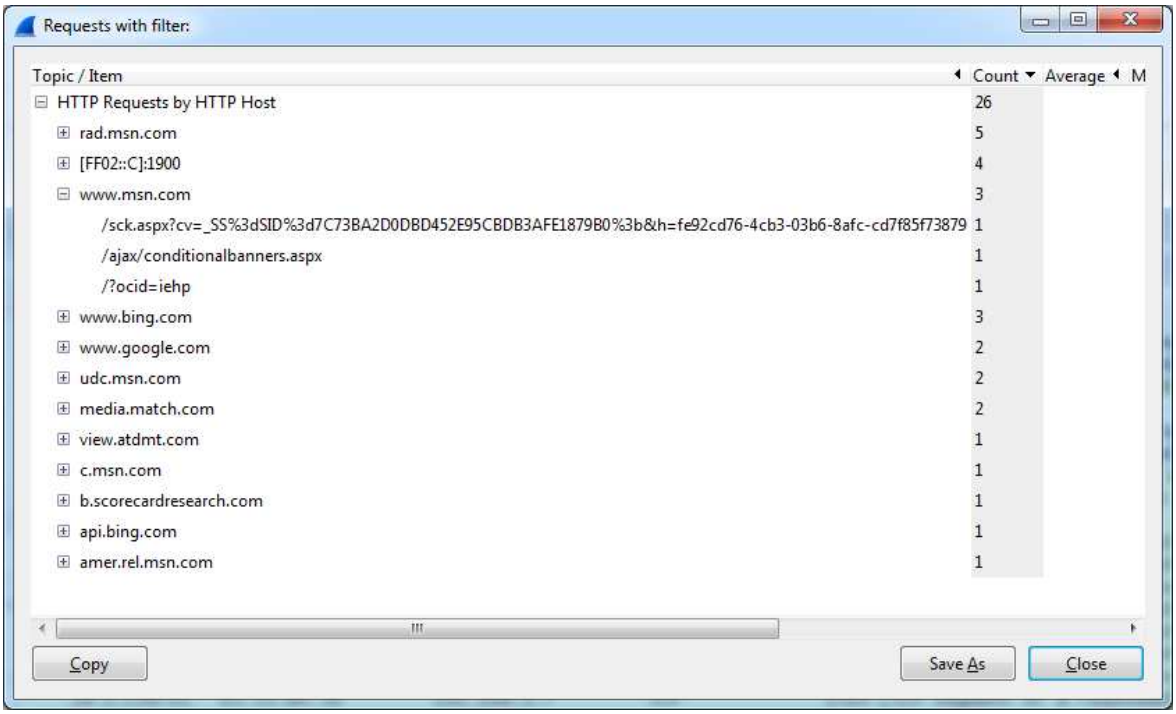
Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	266	100.00 %	141368	0.113	0	0	0.000
Ethernet	100.00 %	266	100.00 %	141368	0.113	0	0	0.000
Internet Protocol Version 6	1.50 %	4	0.59 %	832	0.001	0	0	0.000
User Datagram Protocol	1.50 %	4	0.59 %	832	0.001	0	0	0.000
Hypertext Transfer Protocol	1.50 %	4	0.59 %	832	0.001	4	832	0.001
Internet Protocol Version 4	97.37 %	259	99.30 %	140374	0.112	0	0	0.000
User Datagram Protocol	10.15 %	27	2.06 %	2918	0.002	0	0	0.000
Domain Name Service	9.02 %	24	1.85 %	2617	0.002	24	2617	0.002
Data	1.13 %	3	0.21 %	301	0.000	3	301	0.000
Transmission Control Protocol	86.84 %	231	96.16 %	135942	0.109	177	106781	0.085
Hypertext Transfer Protocol	17.67 %	47	18.57 %	26258	0.021	26	16132	0.013
Line-based text data	4.51 %	12	4.58 %	6469	0.005	12	6469	0.005
Compuserve GIF	2.63 %	7	2.21 %	3119	0.002	7	3119	0.002
JavaScript Object Notation	0.75 %	2	0.38 %	538	0.000	2	538	0.000
Secure Sockets Layer	[illegible]	7	2.05 %	2903	0.002	7	2903	0.002
Data	[illegible]	1	1.07 %	1514	0.001	1	1514	0.001
Address Resolution Protocol	[illegible]	[illegible]	[illegible]	162	0.000	3	162	0.000

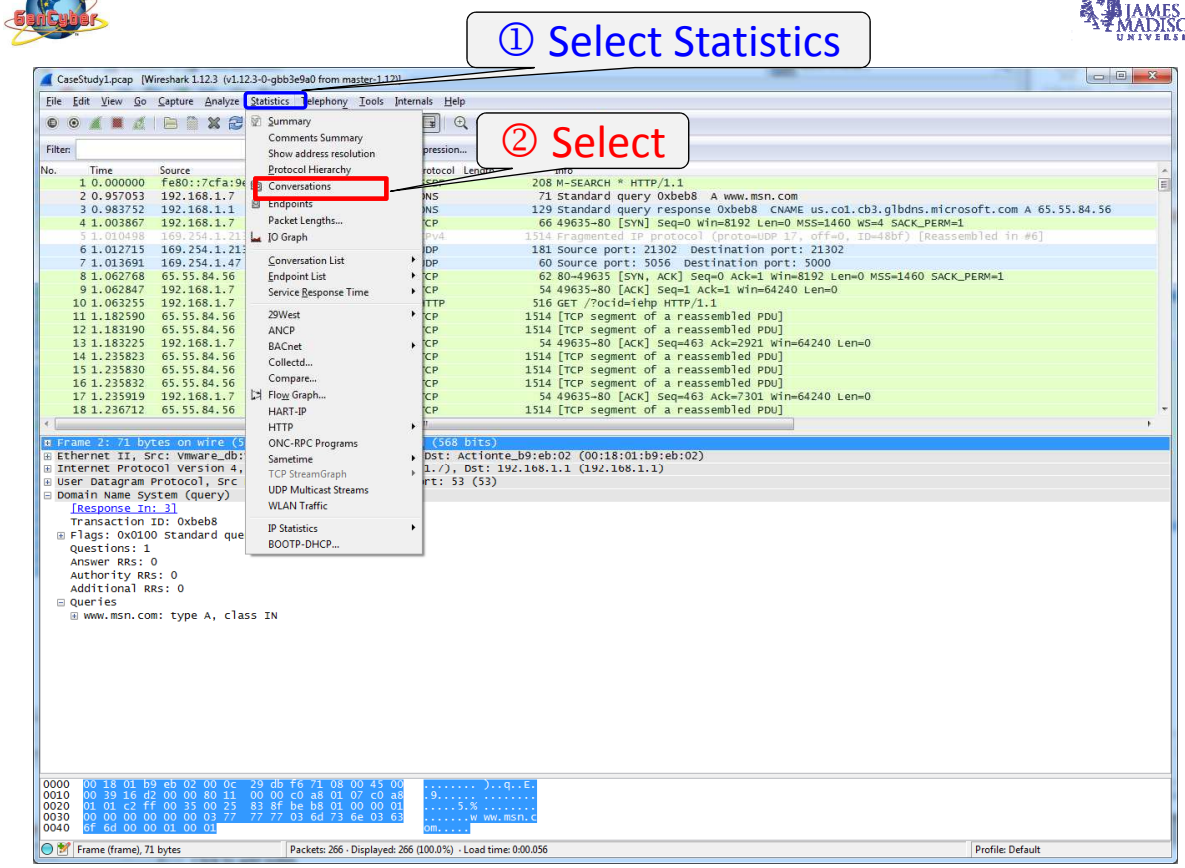

⑥ Select

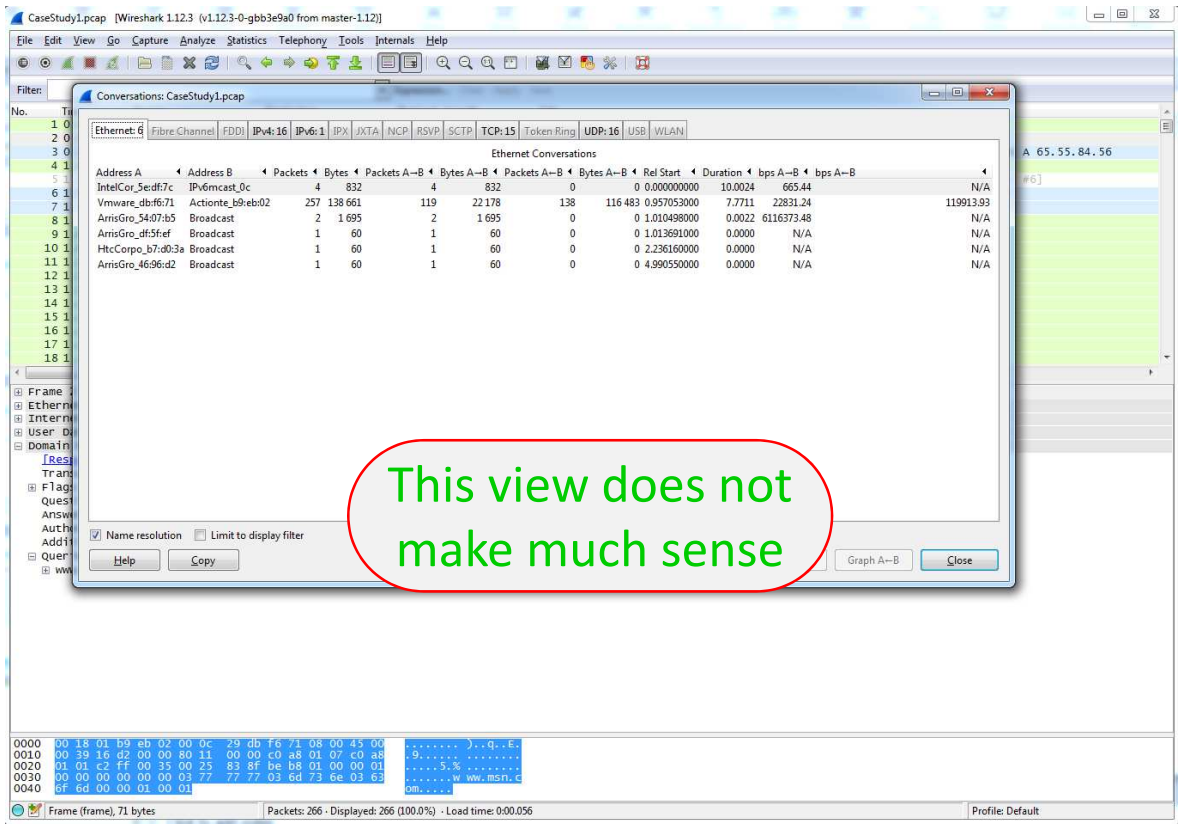

⑦ Select

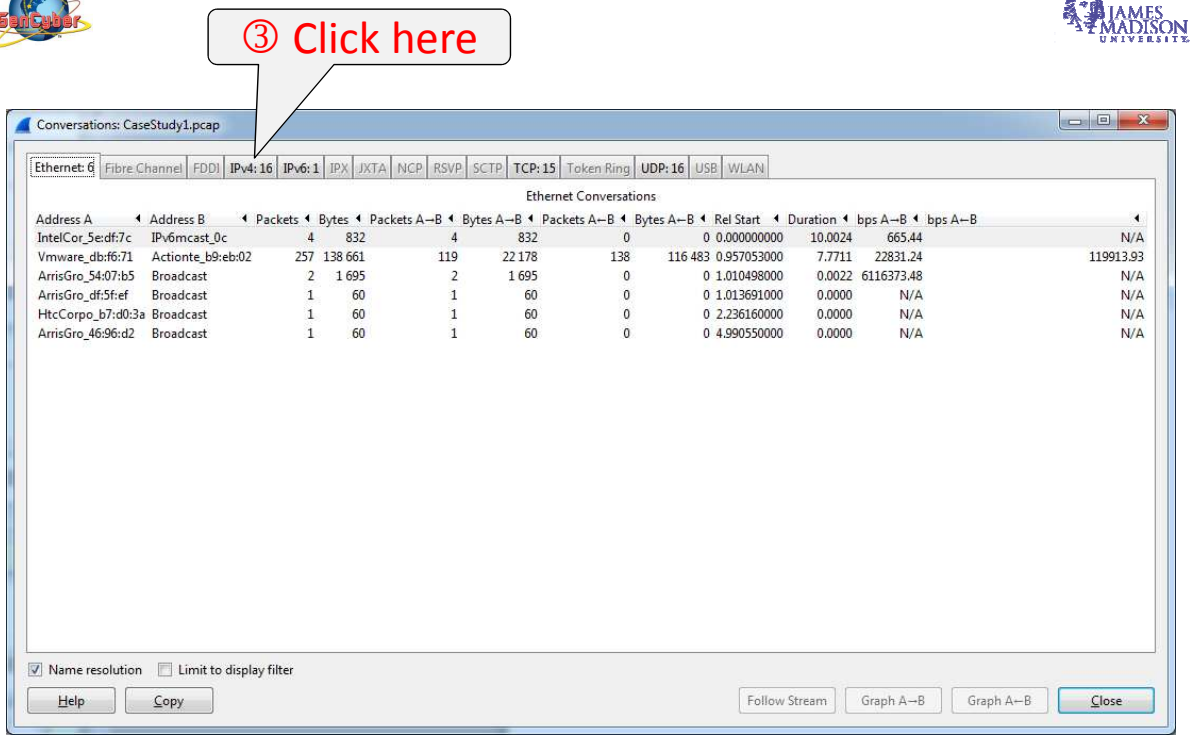

CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Wireshark: Requests Stats Tree...
Filter:
Create Stat
Cancel
⑧ Click


⑨ This shows the distribution of HTTP requests to each server
Requests with filter:
Topic / Item
HTTP Requests by HTTP Host
rad.msn.com
[FF02::C]:1900
www.msn.com
/sck.aspx?cv=_SS%3dSID%3d7C73BA2D0DBD452E95CBDB3AFE1879B0%3b&h=fe92cd76-4cb3-03b6-8afc-cd7f85f73879
/ajax/conditionalbanners.aspx
/?ocid=iehp
www.bing.com
www.google.com
udc.msn.com
media.match.com
view.atdmt.com
c.msn.com
b.scorecardresearch.com
Copy
Save As
Close

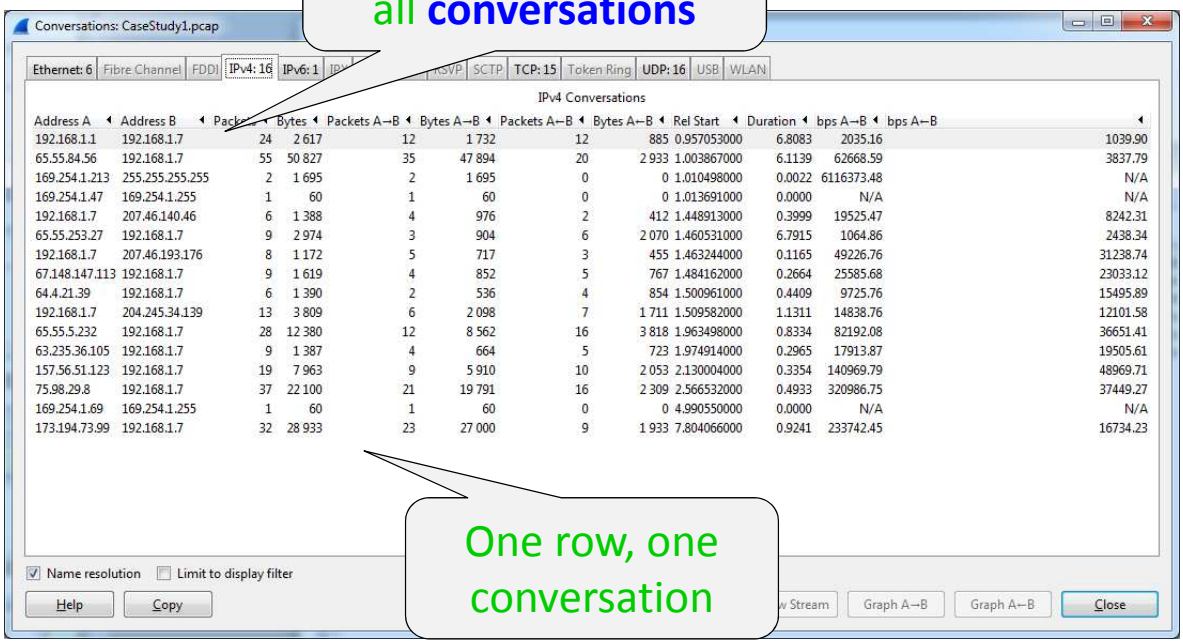

Requests with filter:
Topic / Item
Count
Average
M
HTTP Requests by HTTP Host 26
rad.msn.com 5
[FF02::C]:1900 4
www.msn.com 3
/sck.aspx?cv=_SS%3dSID%3d7C73BA2D0DBD452E95CBDB3AFE1879B0%3b&h=fe92cd76-4cb3-03b6-8afc-cd7f85f73879 1
/ajax/conditionalbanners.aspx 1
/?ocid=iehp 1
www.bing.com 3
www.google.com 2
udc.msn.com 2
media.match.com 2
view.atdmt.com 1
c.msn.com 1
b.scorecardresearch.com 1
api.bing.com 1
amer.rel.msn.com 1
Copy
Save As
Close


① Select Statistics
② Select
CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Summary
Comments Summary
Show address resolution
Protocol Hierarchy
Conversations
Endpoints
Packet Lengths...
IO Graph
Conversation List
Endpoint List
Service Response Time
29West
ANCP
BACnet
Collectd...
Compare...
Flow Graph...
HART-IP
HTTP
ONC-RPC Programs
Sametime
TCP StreamGraph
UDP Multicast Streams
WLAN Traffic
IP Statistics
BOOTP-DHCP...
Frame (frame), 71 bytes
Packets: 266 · Displayed: 266 (100.0%) · Load time: 0:00.056
Profile: Default

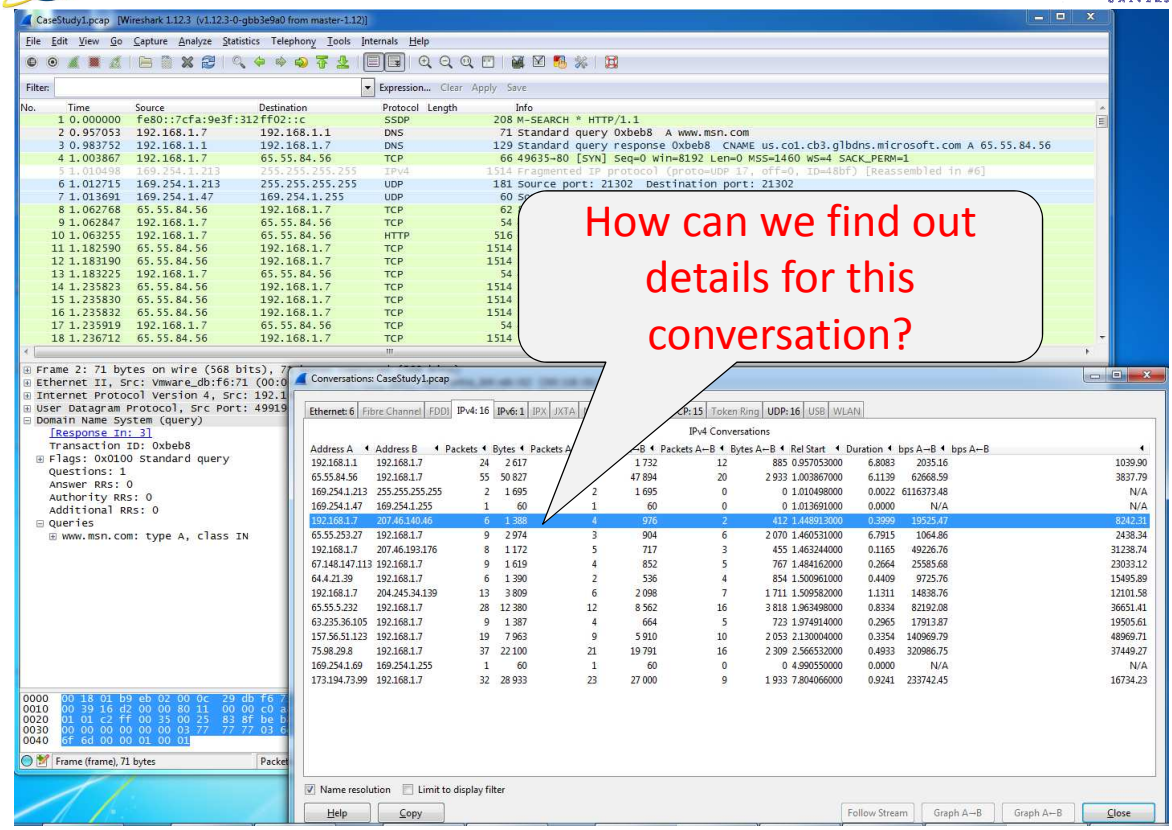

CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Conversations: CaseStudy1.pcap
Ethernet: 6 Fibre Channel FDDI IPv4: 16 IPv6: 1 IPX JXTA NCP RSVP SCTP TCP: 15 Token Ring UDP: 16 USB WLAN
Ethernet Conversations
Address A Address B Packets Bytes Packets A→B Bytes A→B Packets A←B Bytes A←B Rel Start Duration bps A→B bps A←B
IntelCor_5e:df:7c IPv6mcast_0c 4 832 4 832 0 0 0.000000000 10.0024 665.44 N/A
Vmware_db:f6:71 Actionte_b9:eb:02 257 138 661 119 22 178 138 116 483 0.957053000 7.7711 22831.24 119913.93
ArrisGro_54:07:b5 Broadcast 2 1 695 2 1 695 0 0 1.010498000 0.0022 6116373.48 N/A
ArrisGro_df:5f:ef Broadcast 1 60 1 60 0 0 1.013691000 0.0000 N/A N/A
HtcCorpo_b7:d0:3a Broadcast 1 60 1 60 0 0 2.236160000 0.0000 N/A N/A
ArrisGro_46:96:d2 Broadcast 1 60 1 60 0 0 4.990550000 0.0000 N/A N/A
Name resolution Limit to display filter
Help Copy Graph A→B Close
This view does not make much sense
Frame (frame), 71 bytes
Packets: 266 · Displayed: 266 (100.0%) · Load time: 0:00.056
Profile: Default

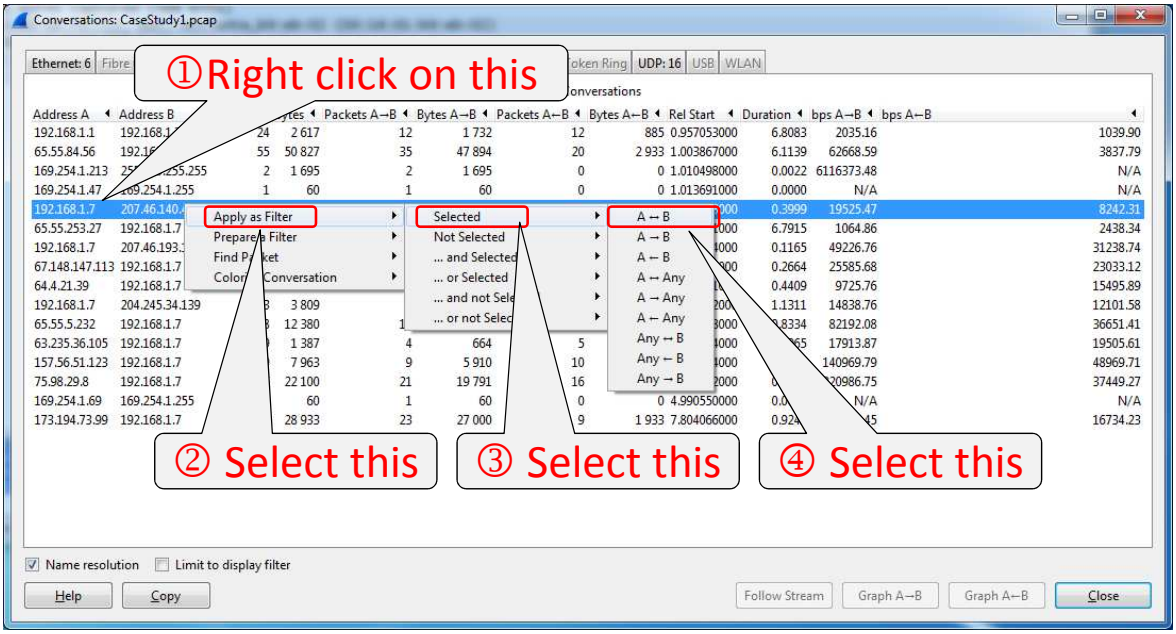

③ Click here
Conversations: CaseStudy1.pcap
Ethernet: 6 Fibre Channel FDDI IPv4: 16 IPv6: 1 IPX JXTA NCP RSVP SCTP TCP: 15 Token Ring UDP: 16 USB WLAN
Ethernet Conversations
Address A Address B Packets Bytes Packets A→B Bytes A→B Packets A←B Bytes A←B Rel Start Duration bps A→B bps A←B
IntelCor_5e:df:7c IPv6mcast_0c 4 832 4 832 0 0 0.000000000 10.0024 665.44 N/A
Vmware_db:f6:71 Actionte_b9:eb:02 257 138 661 119 22 178 138 116 483 0.957053000 7.7711 22831.24 119913.93
ArrisGro_54:07:b5 Broadcast 2 1 695 2 1 695 0 0 1.010498000 0.0022 6116373.48 N/A
ArrisGro_df:5f:ef Broadcast 1 60 1 60 0 0 1.013691000 0.0000 N/A N/A
HtcCorpo_b7:d0:3a Broadcast 1 60 1 60 0 0 2.236160000 0.0000 N/A N/A
ArrisGro_46:96:d2 Broadcast 1 60 1 60 0 0 4.990550000 0.0000 N/A N/A
Name resolution Limit to display filter
Help Copy Follow Stream Graph A→B Graph A←B Close

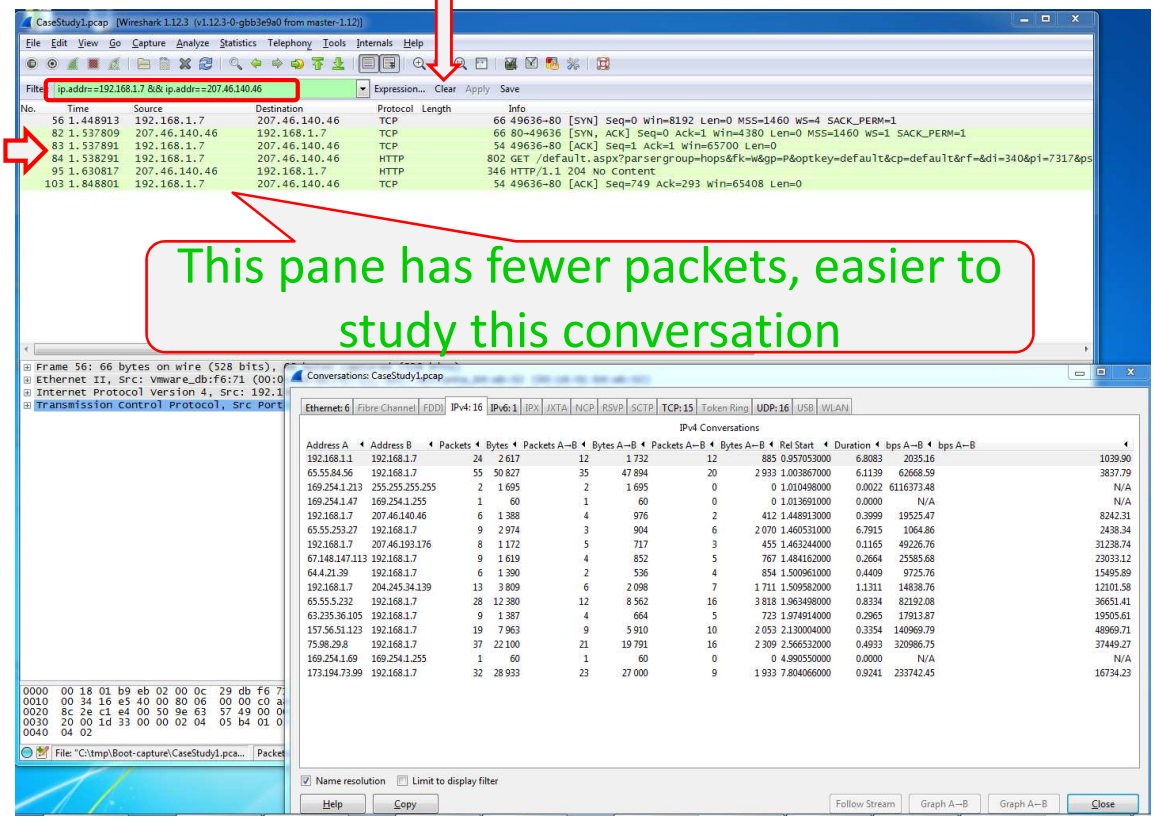
④ This window shows all **conversations**

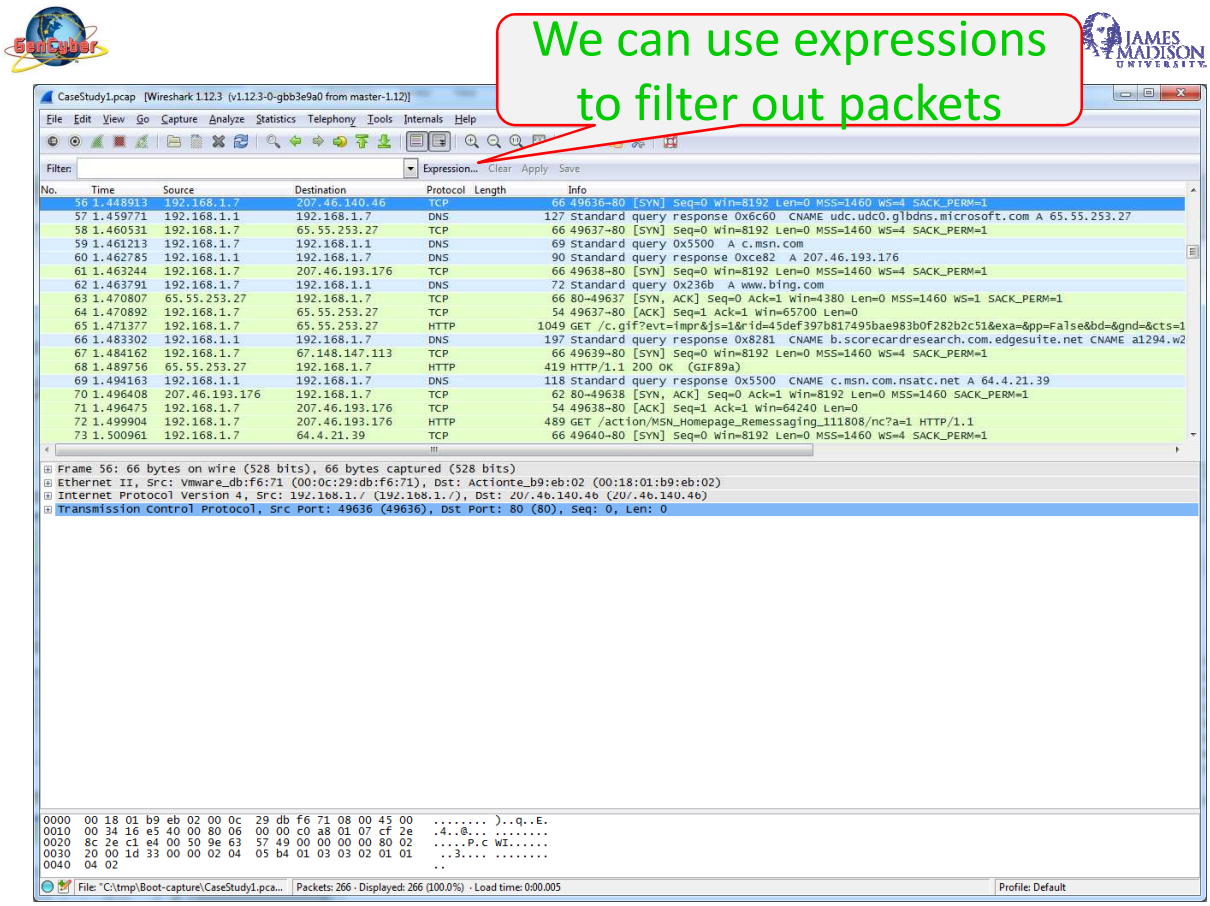


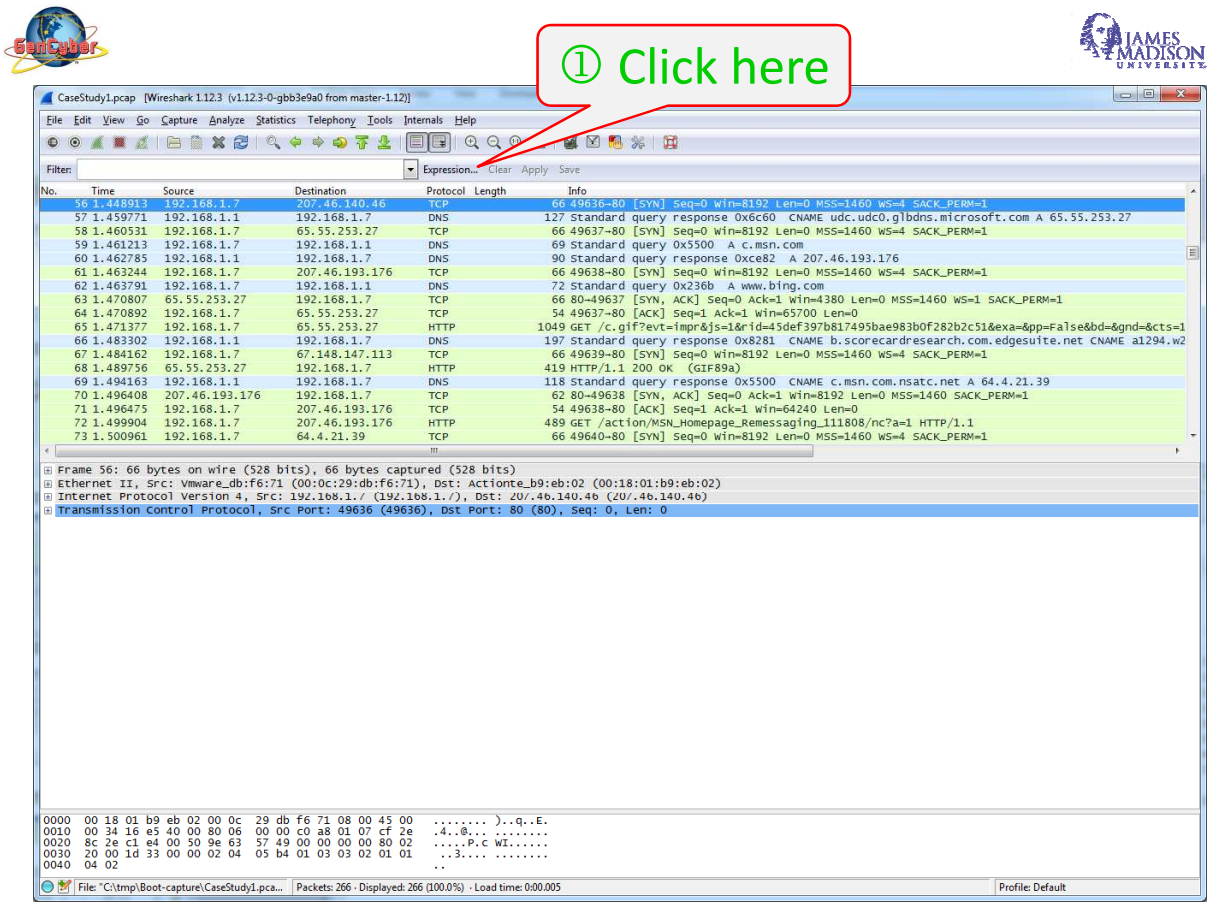
One row, one conversation

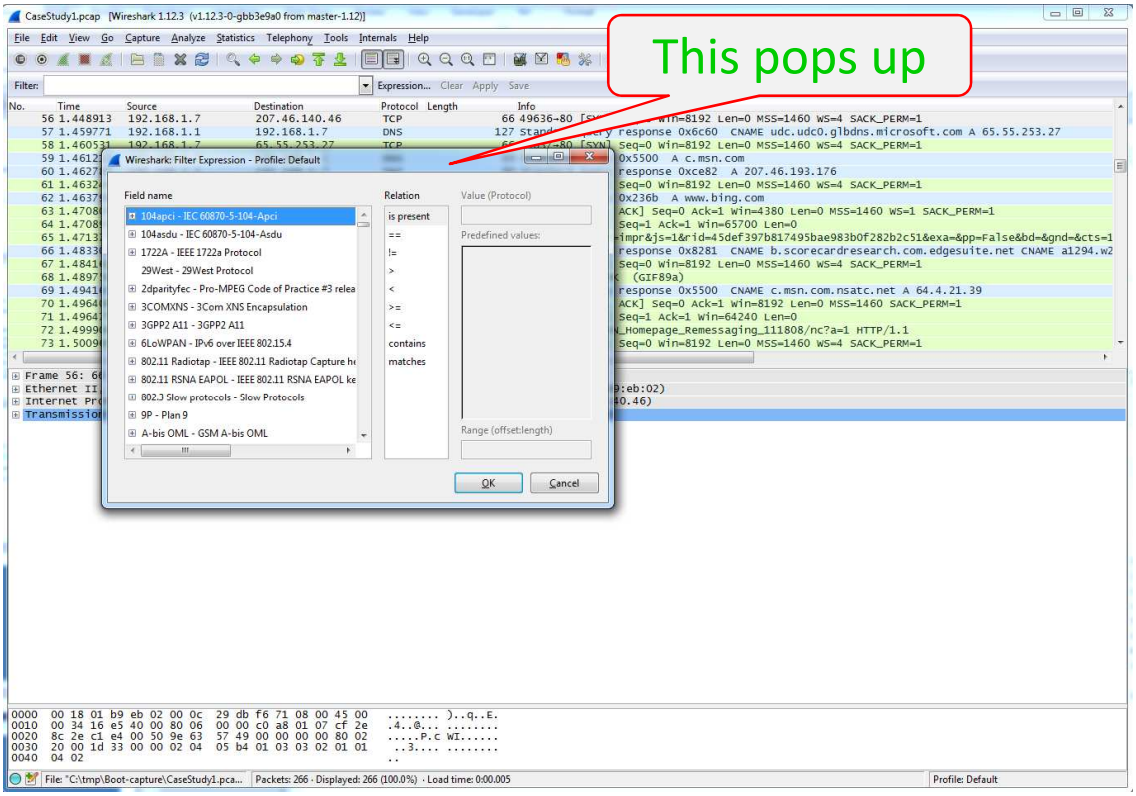


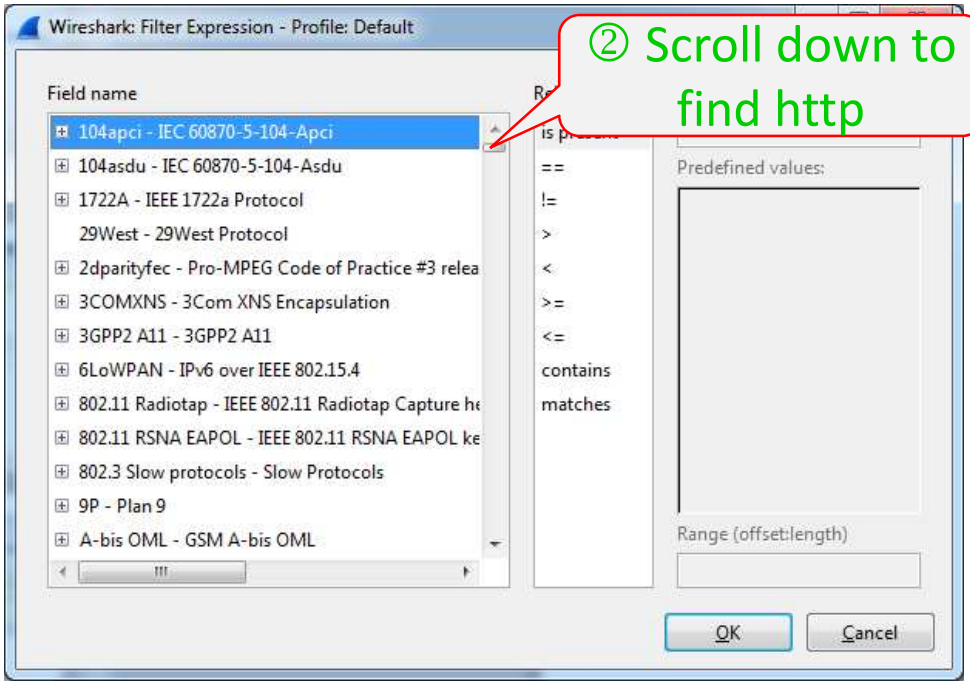

Conversations: CaseStudy1.pcap
①Right click on this
Apply as Filter
Prepare a Filter
Find Packet
Color Conversation
Selected
Not Selected
... and Selected
... or Selected
... and not Selected
... or not Selected
A ↔ B
A → B
A ← B
A ↔ Any
A → Any
A ← Any
Any ↔ B
Any ← B
Any → B
② Select this
③ Select this
④ Select this
Name resolution
Limit to display filter
Help
Copy
Follow Stream
Graph A→B
Graph A←B
Close

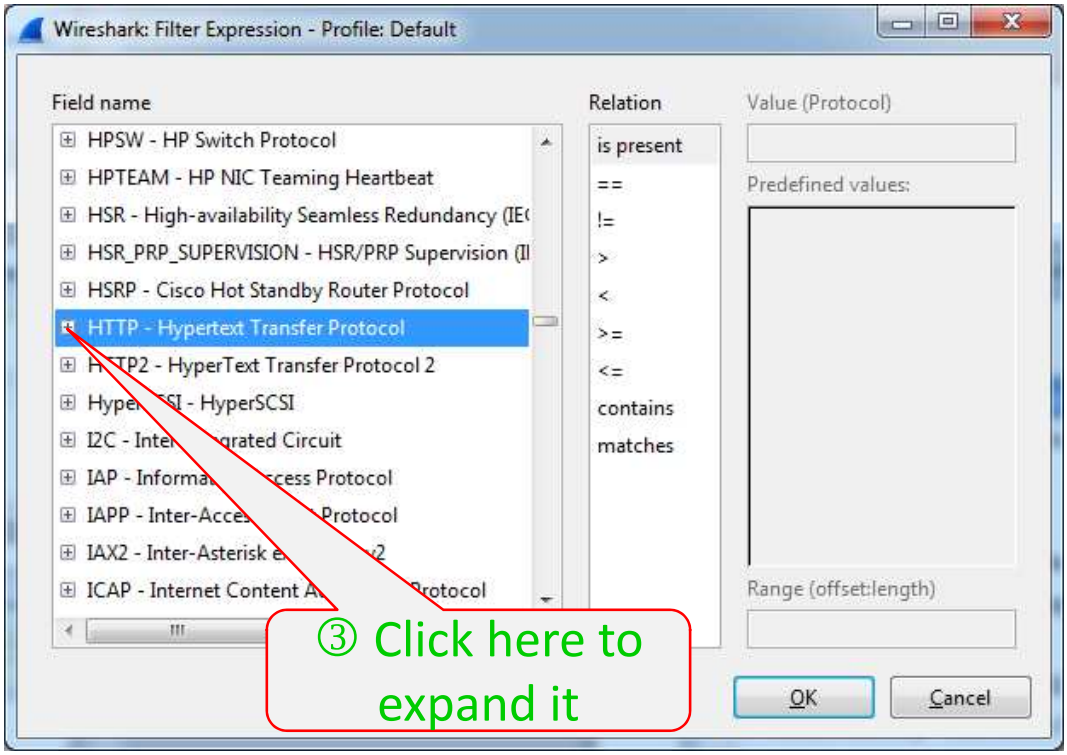

CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
ip.addr==192.168.1.7 && ip.addr==207.46.140.46
This pane has fewer packets, easier to study this conversation
Conversations: CaseStudy1.pcap
IPv4 Conversations

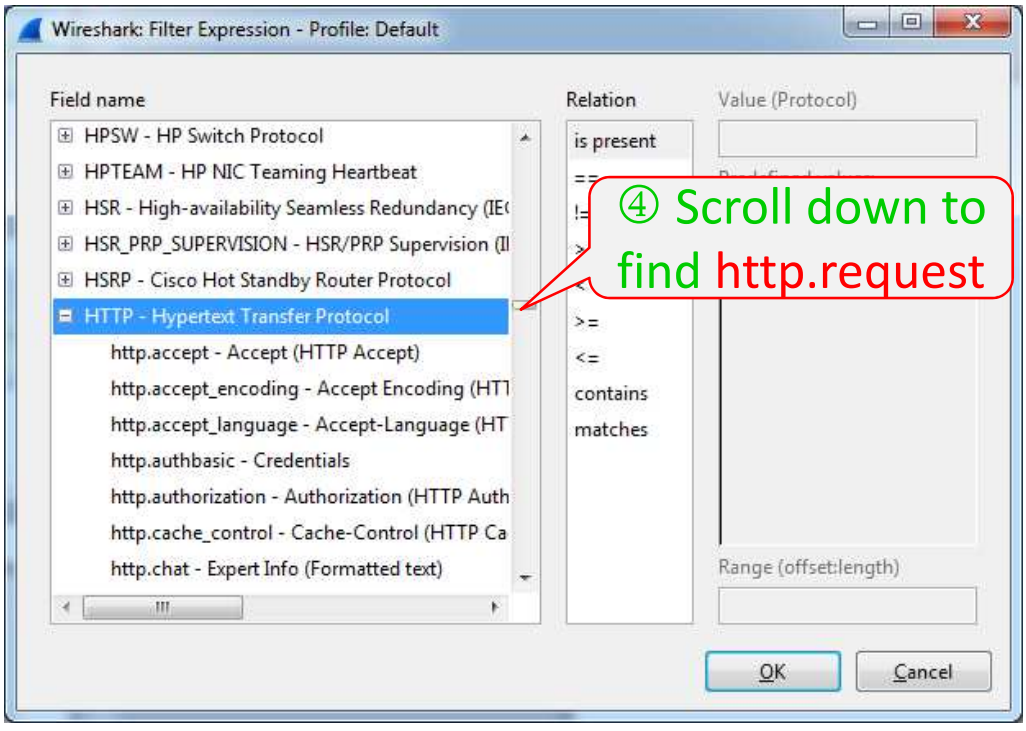

We can use expressions to filter out packets
CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
56 1.448913 192.168.1.7 207.46.140.46 TCP 66 49636→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
57 1.459771 192.168.1.1 192.168.1.7 DNS 127 Standard query response 0x6c60 CNAME udc.udc0.glbdns.microsoft.com A 65.55.253.27
58 1.460531 192.168.1.7 65.55.253.27 TCP 66 49637→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
59 1.461213 192.168.1.7 192.168.1.1 DNS 69 Standard query 0x5500 A c.msn.com
60 1.462785 192.168.1.1 192.168.1.7 DNS 90 Standard query response 0xce82 A 207.46.193.176
61 1.463244 192.168.1.7 207.46.193.176 TCP 66 49638→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
62 1.463791 192.168.1.7 192.168.1.1 DNS 72 Standard query 0x236b A www.bing.com
63 1.470807 65.55.253.27 192.168.1.7 TCP 66 80→49637 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 WS=1 SACK_PERM=1
64 1.470892 192.168.1.7 65.55.253.27 TCP 54 49637→80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
65 1.471377 192.168.1.7 65.55.253.27 HTTP 1049 GET /c.gif?evt=impr&js=1&rid=45def397b817495bae983b0f282b2c51&exa=&pp=False&bd=&gnd=&cts=1
66 1.483302 192.168.1.1 192.168.1.7 DNS 197 Standard query response 0x8281 CNAME b.scorecardresearch.com.edgesuite.net CNAME a1294.w2
67 1.484162 192.168.1.7 67.148.147.113 TCP 66 49639→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
68 1.489756 65.55.253.27 192.168.1.7 HTTP 419 HTTP/1.1 200 OK (GIF89a)
69 1.494163 192.168.1.1 192.168.1.7 DNS 118 Standard query response 0x5500 CNAME c.msn.com.nsatc.net A 64.4.21.39
70 1.496408 207.46.193.176 192.168.1.7 TCP 62 80→49638 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
71 1.496475 192.168.1.7 207.46.193.176 TCP 54 49638→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
72 1.499904 192.168.1.7 207.46.193.176 HTTP 489 GET /action/MSN_Homepage_Remessaging_111808/nc?a=1 HTTP/1.1
73 1.500961 192.168.1.7 64.4.21.39 TCP 66 49640→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
Frame 56: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Vmware_db:f6:71 (00:0c:29:db:f6:71), Dst: Actionte_b9:eb:02 (00:18:01:b9:eb:02)
Internet Protocol Version 4, Src: 192.168.1.7 (192.168.1.7), Dst: 207.46.140.46 (207.46.140.46)
Transmission Control Protocol, Src Port: 49636 (49636), Dst Port: 80 (80), Seq: 0, Len: 0
0000 00 18 01 b9 eb 02 00 0c 29 db f6 71 08 00 45 00 ....)..q..E.
0010 00 34 16 e5 40 00 80 06 00 00 c0 a8 01 07 cf 2e .4..@... ....
0020 8c 2e c1 e4 00 50 9e 63 57 49 00 00 00 00 80 02 .....P.c WI......
0030 20 00 1d 33 00 00 02 04 05 b4 01 03 03 02 01 01 ..3.... ....
0040 04 02 ..
File: "C:\tmp\Boot-capture\CaseStudy1.pca... Packets: 266 · Displayed: 266 (100.0%) · Load time: 0:00.005
Profile: Default

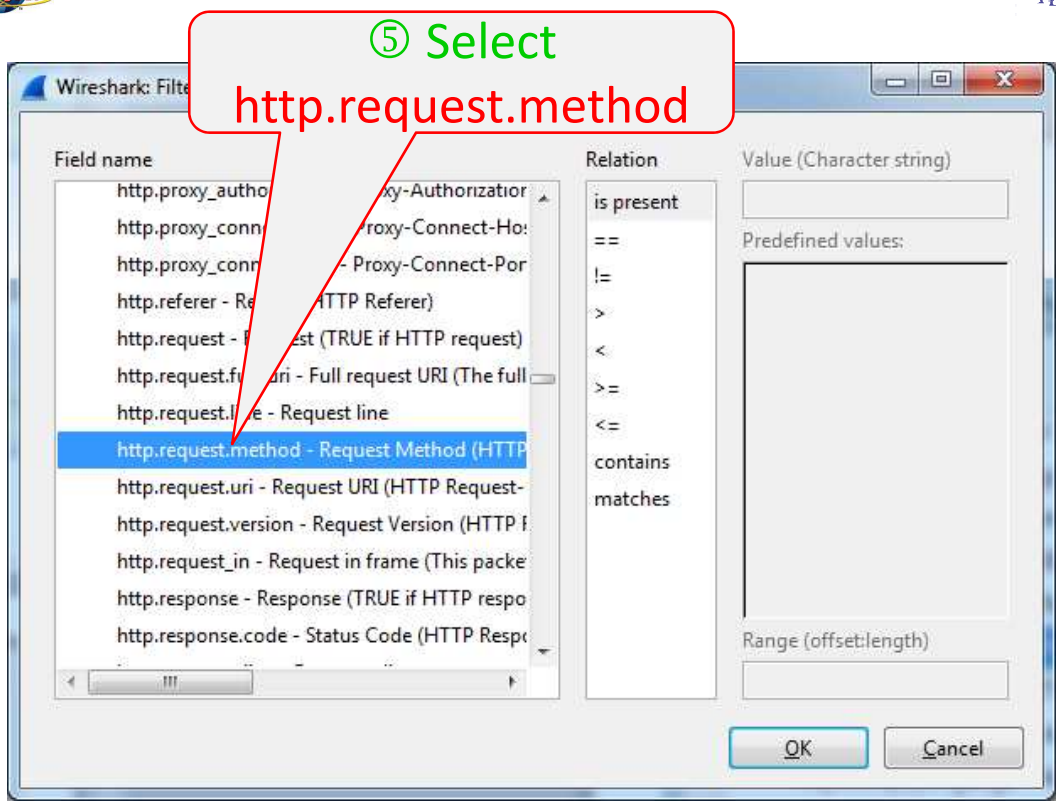

① Click here
CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
56 1.448913 192.168.1.7 207.46.140.46 TCP 66 49636→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
57 1.459771 192.168.1.1 192.168.1.7 DNS 127 Standard query response 0x6c60 CNAME udc.udc0.glbdns.microsoft.com A 65.55.253.27
58 1.460531 192.168.1.7 65.55.253.27 TCP 66 49637→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
59 1.461213 192.168.1.7 192.168.1.1 DNS 69 Standard query 0x5500 A c.msn.com
60 1.462785 192.168.1.1 192.168.1.7 DNS 90 Standard query response 0xce82 A 207.46.193.176
61 1.463244 192.168.1.7 207.46.193.176 TCP 66 49638→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
62 1.463791 192.168.1.7 192.168.1.1 DNS 72 Standard query 0x236b A www.bing.com
63 1.470807 65.55.253.27 192.168.1.7 TCP 66 80→49637 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 WS=1 SACK_PERM=1
64 1.470892 192.168.1.7 65.55.253.27 TCP 54 49637→80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
65 1.471377 192.168.1.7 65.55.253.27 HTTP 1049 GET /c.gif?evt=impr&js=1&rid=45def397b817495bae983b0f282b2c51&exa=&pp=False&bd=&gnd=&cts=1
66 1.483302 192.168.1.1 192.168.1.7 DNS 197 Standard query response 0x8281 CNAME b.scorecardresearch.com.edgesuite.net CNAME a1294.w2
67 1.484162 192.168.1.7 67.148.147.113 TCP 66 49639→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
68 1.489756 65.55.253.27 192.168.1.7 HTTP 419 HTTP/1.1 200 OK (GIF89a)
69 1.494163 192.168.1.1 192.168.1.7 DNS 118 Standard query response 0x5500 CNAME c.msn.com.nsatc.net A 64.4.21.39
70 1.496408 207.46.193.176 192.168.1.7 TCP 62 80→49638 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 SACK_PERM=1
71 1.496475 192.168.1.7 207.46.193.176 TCP 54 49638→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
72 1.499904 192.168.1.7 207.46.193.176 HTTP 489 GET /action/MSN_Homepage_Remessaging_111808/nc?a=1 HTTP/1.1
73 1.500961 192.168.1.7 64.4.21.39 TCP 66 49640→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
Frame 56: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: Vmware_db:f6:71 (00:0c:29:db:f6:71), Dst: Actionte_b9:eb:02 (00:18:01:b9:eb:02)
Internet Protocol Version 4, Src: 192.168.1.7 (192.168.1.7), Dst: 207.46.140.46 (207.46.140.46)
Transmission Control Protocol, Src Port: 49636 (49636), Dst Port: 80 (80), Seq: 0, Len: 0
0000 00 18 01 b9 eb 02 00 0c 29 db f6 71 08 00 45 00 ....)..q..E.
0010 00 34 16 e5 40 00 80 06 00 00 c0 a8 01 07 cf 2e .4..@... ....
0020 8c 2e c1 e4 00 50 9e 63 57 49 00 00 00 00 80 02 .....P.c WI......
0030 20 00 1d 33 00 00 02 04 05 b4 01 03 03 02 01 01 ..3.... ....
0040 04 02 ..
File: "C:\tmp\Boot-capture\CaseStudy1.pca... Packets: 266 · Displayed: 266 (100.0%) · Load time: 0:00.005
Profile: Default

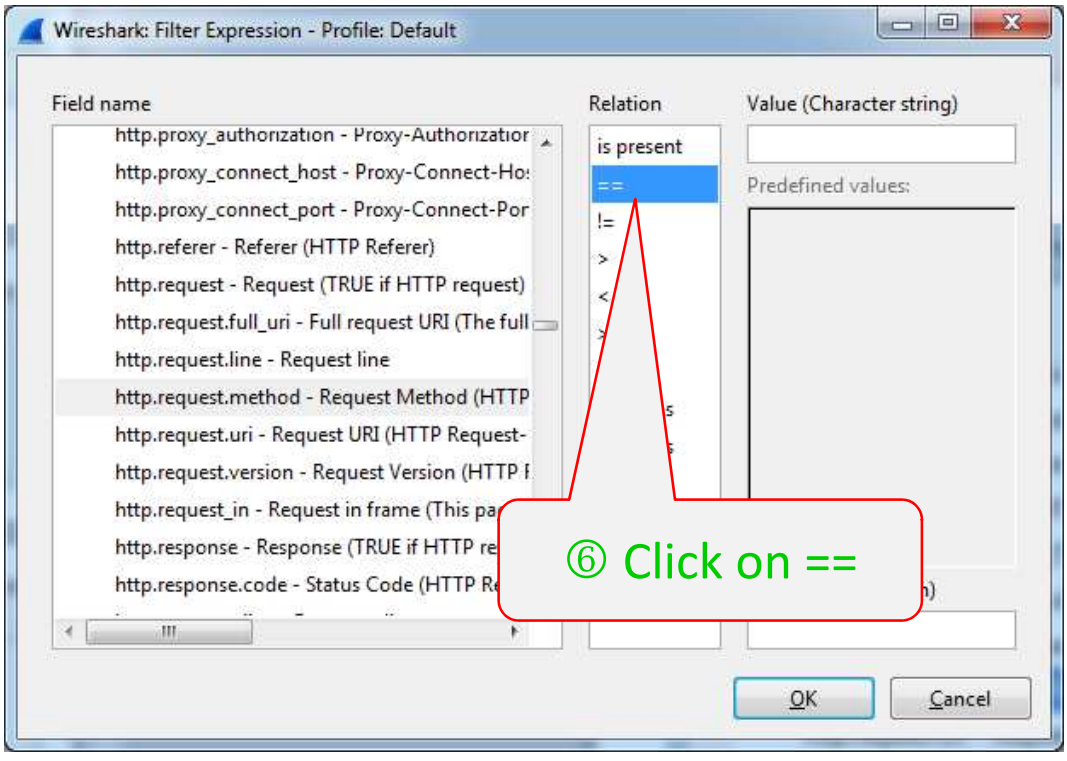

This pops up
CaseStudy1.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Wireshark: Filter Expression - Profile: Default
Field name
Relation
Value (Protocol)
Predefined values:
Range (offset:length)
OK
Cancel

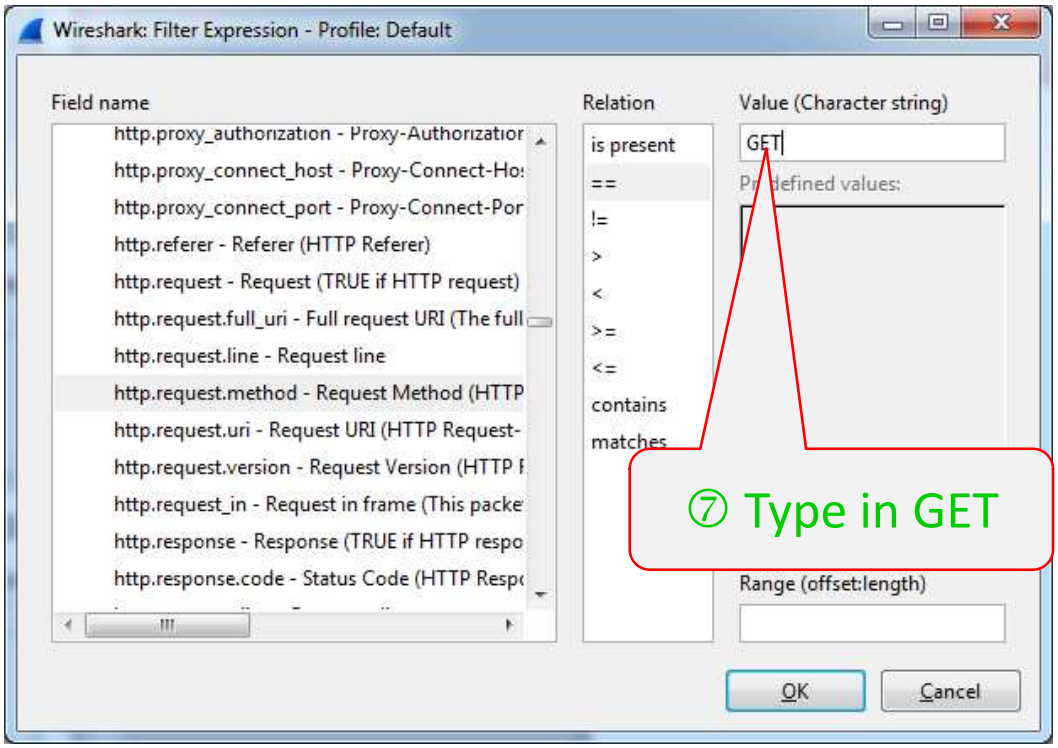

② Scroll down to find http
Wireshark: Filter Expression - Profile: Default
Field name
104apci - IEC 60870-5-104-Apci
104asdu - IEC 60870-5-104-Asdu
1722A - IEEE 1722a Protocol
29West - 29West Protocol
2dparityfec - Pro-MPEG Code of Practice #3 relea
3COMXNS - 3Com XNS Encapsulation
3GPP2 A11 - 3GPP2 A11
6LoWPAN - IPv6 over IEEE 802.15.4
802.11 Radiotap - IEEE 802.11 Radiotap Capture he
802.11 RSNA EAPOL - IEEE 802.11 RSNA EAPOL ke
802.3 Slow protocols - Slow Protocols
9P - Plan 9
A-bis OML - GSM A-bis OML
==
!=
>
<
>=
<=
contains
matches
Predefined values:
Range (offset:length)
OK
Cancel

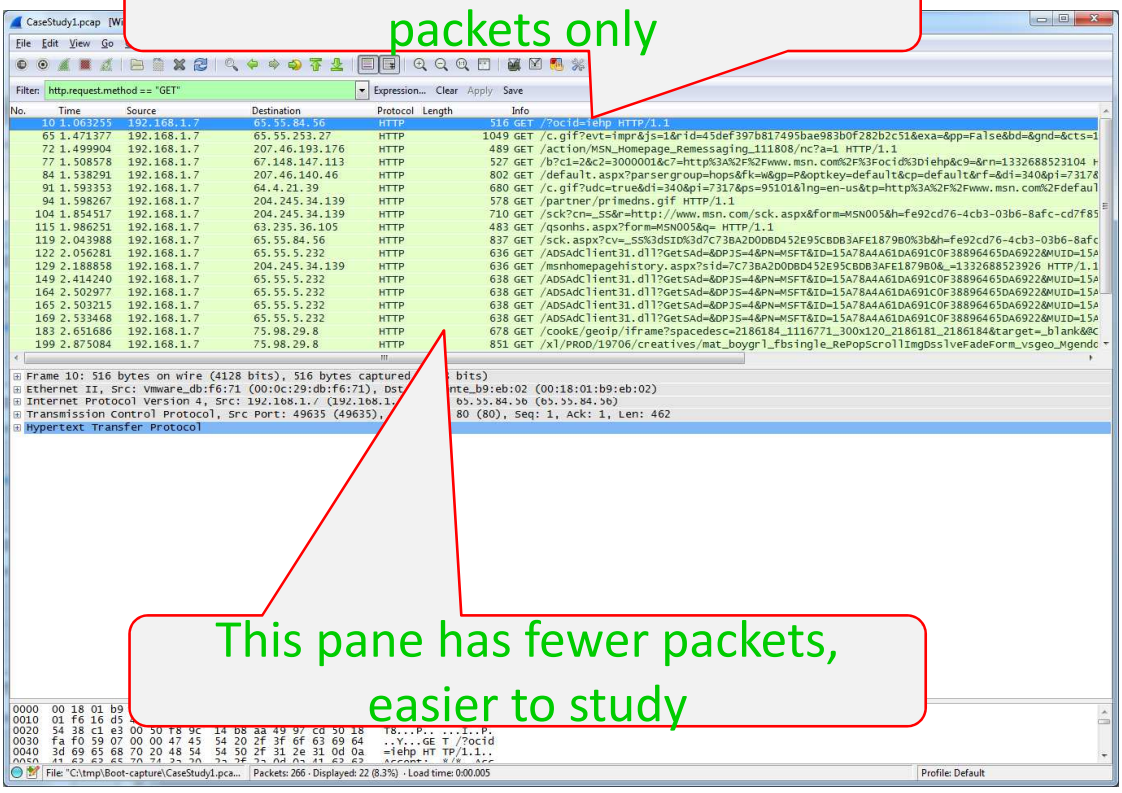

Wireshark: Filter Expression - Profile: Default
Field name
Relation
Value (Protocol)
HPSW - HP Switch Protocol
HPTEAM - HP NIC Teaming Heartbeat
HSR - High-availability Seamless Redundancy (IE
HSR_PRP_SUPERVISION - HSR/PRP Supervision (I
HSRP - Cisco Hot Standby Router Protocol
HTTP - Hypertext Transfer Protocol
HTTP2 - HyperText Transfer Protocol 2
HyperSCSI - HyperSCSI
I2C - Inte grated Circuit
IAP - Informa cess Protocol
IAPP - Inter-Acces Protocol
IAX2 - Inter-Asterisk e 2
ICAP - Internet Content A Protocol
is present
==
!=
>
<
>=
<=
contains
matches
Predefined values:
Range (offset:length)
③ Click here to expand it
OK
Cancel


Wireshark: Filter Expression - Profile: Default
Field name
Relation
Value (Protocol)
HPSW - HP Switch Protocol
HPTEAM - HP NIC Teaming Heartbeat
HSR - High-availability Seamless Redundancy (IE
HSR_PRP_SUPERVISION - HSR/PRP Supervision (I
HSRP - Cisco Hot Standby Router Protocol
HTTP - Hypertext Transfer Protocol
http.accept - Accept (HTTP Accept)
http.accept_encoding - Accept Encoding (HT
http.accept_language - Accept-Language (HT
http.authbasic - Credentials
http.authorization - Authorization (HTTP Auth
http.cache_control - Cache-Control (HTTP Ca
http.chat - Expert Info (Formatted text)
is present
==
!=
>
<
>=
<=
contains
matches
Range (offset:length)
④ Scroll down to find http.request
OK
Cancel

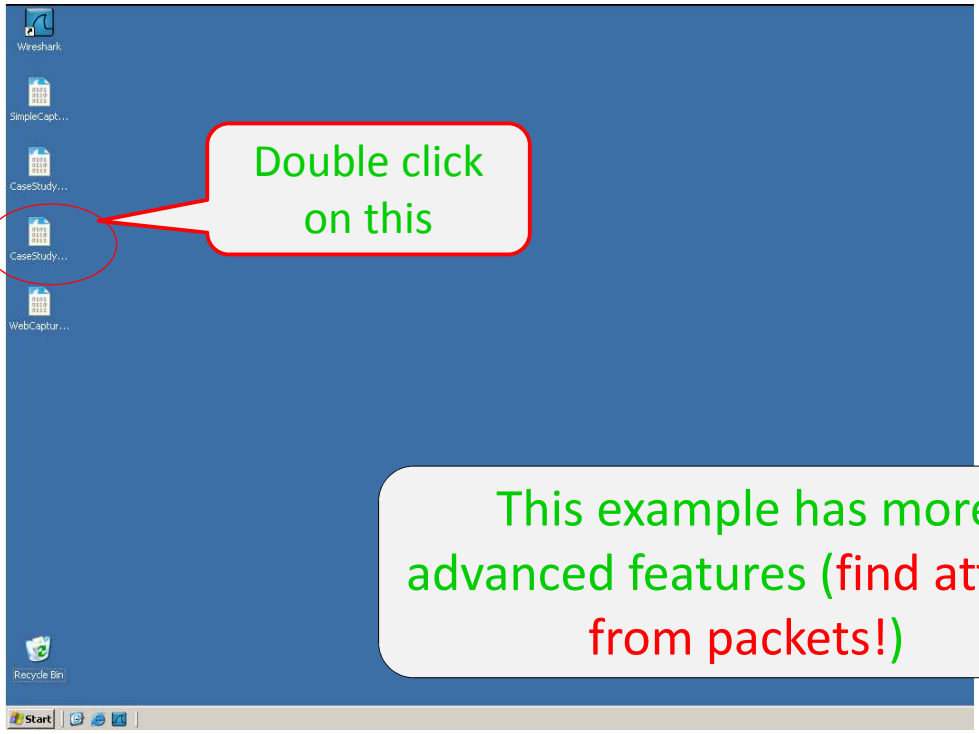

⑤ Select
http.request.method
Wireshark: Filte
Field name
Relation
Value (Character string)
http.proxy_autho
xy-Authorizatior
http.proxy_conn
roxy-Connect-Ho
http.proxy_conn
- Proxy-Connect-Por
http.referer - Re
TTP Referer)
http.request -
st (TRUE if HTTP request)
http.request.fu
ri - Full request URI (The full
http.request.l
e - Request line
http.request.method - Request Method (HTTP
http.request.uri - Request URI (HTTP Request-
http.request.version - Request Version (HTTP
http.request_in - Request in frame (This packe
http.response - Response (TRUE if HTTP respo
http.response.code - Status Code (HTTP Resp
is present
==
!=
>
<
>=
<=
contains
matches
Predefined values:
Range (offset:length)
OK
Cancel

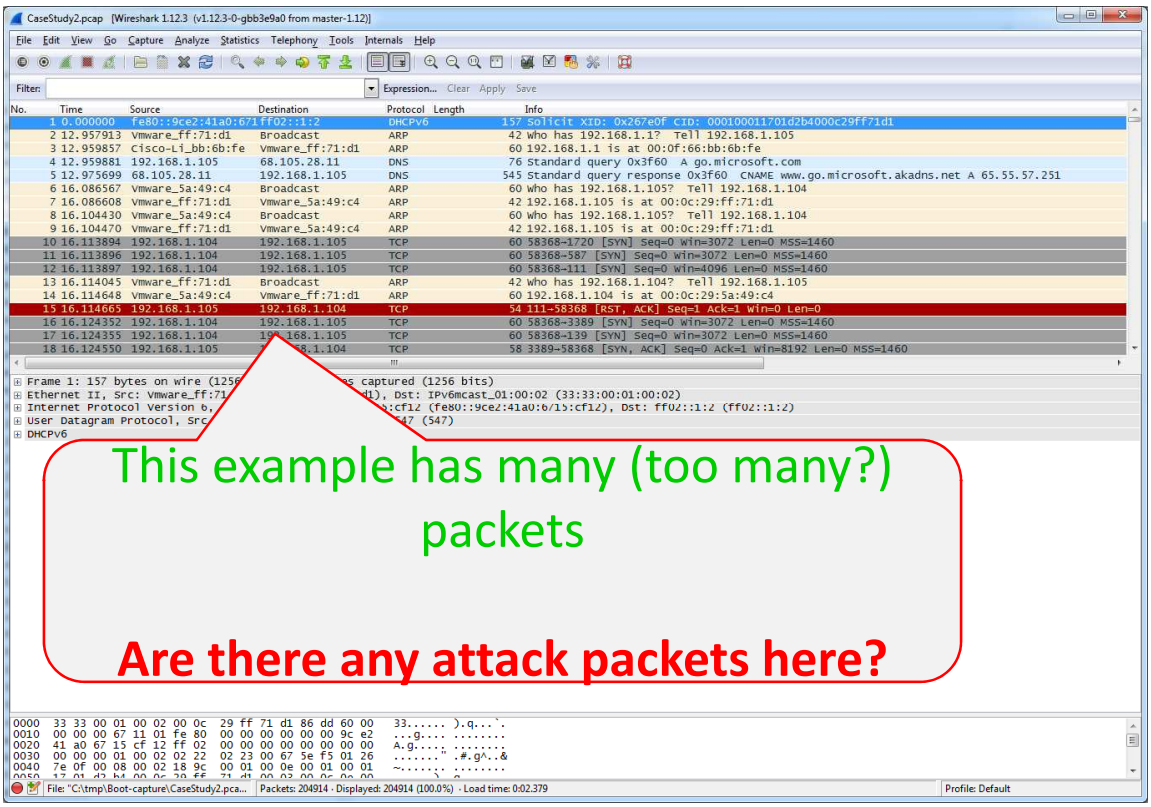

Wireshark: Filter Expression - Profile: Default
Field name
Relation
Value (Character string)
http.proxy_authorization - Proxy-Authorizatior
http.proxy_connect_host - Proxy-Connect-Ho
http.proxy_connect_port - Proxy-Connect-Por
http.referer - Referer (HTTP Referer)
http.request - Request (TRUE if HTTP request)
http.request.full_uri - Full request URI (The full
http.request.line - Request line
http.request.method - Request Method (HTTP
http.request.uri - Request URI (HTTP Request-
http.request.version - Request Version (HTTP
http.request_in - Request in frame (This pa
http.response - Response (TRUE if HTTP re
http.response.code - Status Code (HTTP R
is present
==
!=
>
<
Predefined values:
⑥ Click on ==
OK
Cancel

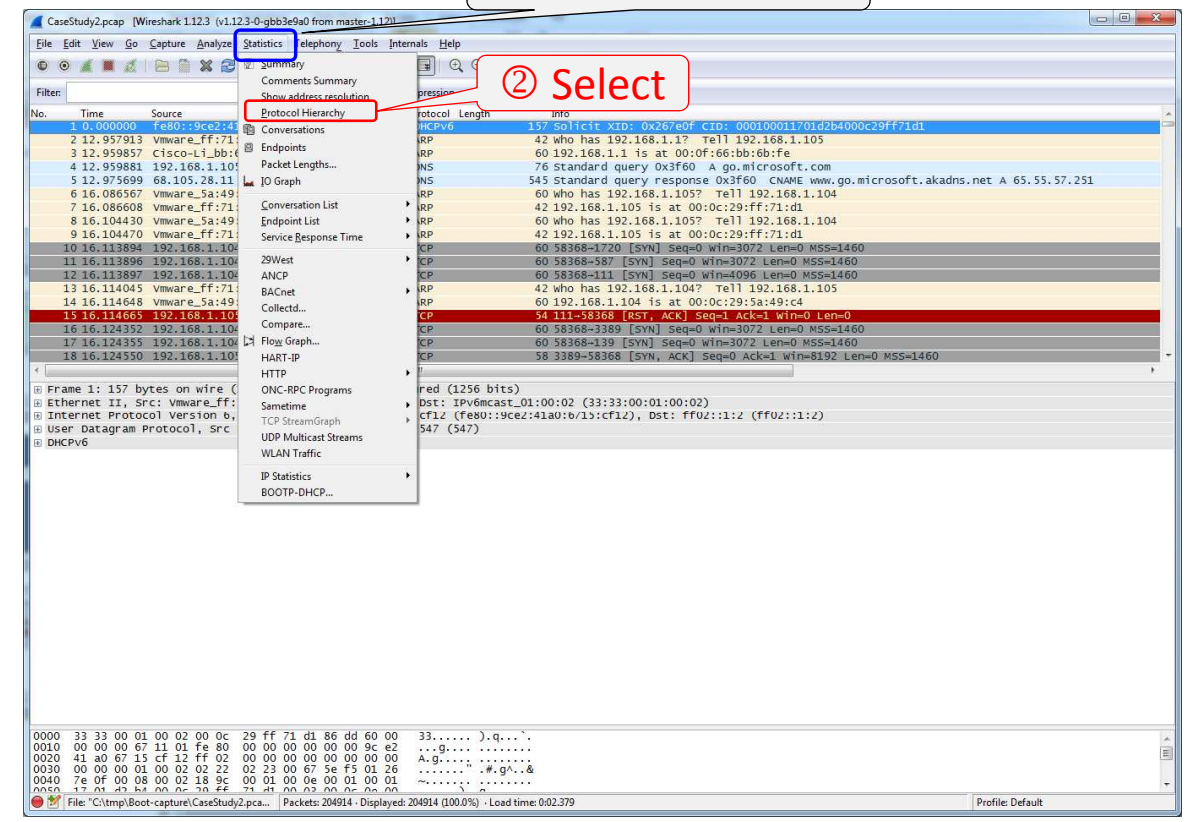

Wireshark: Filter Expression - Profile: Default
Field name
Relation
Value (Character string)
http.proxy_authorization - Proxy-Authorizatior
http.proxy_connect_host - Proxy-Connect-Ho:
http.proxy_connect_port - Proxy-Connect-Por
http.referer - Referer (HTTP Referer)
http.request - Request (TRUE if HTTP request)
http.request.full_uri - Full request URI (The full
http.request.line - Request line
http.request.method - Request Method (HTTP
http.request.uri - Request URI (HTTP Request-
http.request.version - Request Version (HTTP F
http.request_in - Request in frame (This packe
http.response - Response (TRUE if HTTP respo
http.response.code - Status Code (HTTP Resp
is present
==
!=
>
<
>=
<=
contains
matches
GET
Predefined values:
⑦ Type in GET
Range (offset:length)
OK
Cancel

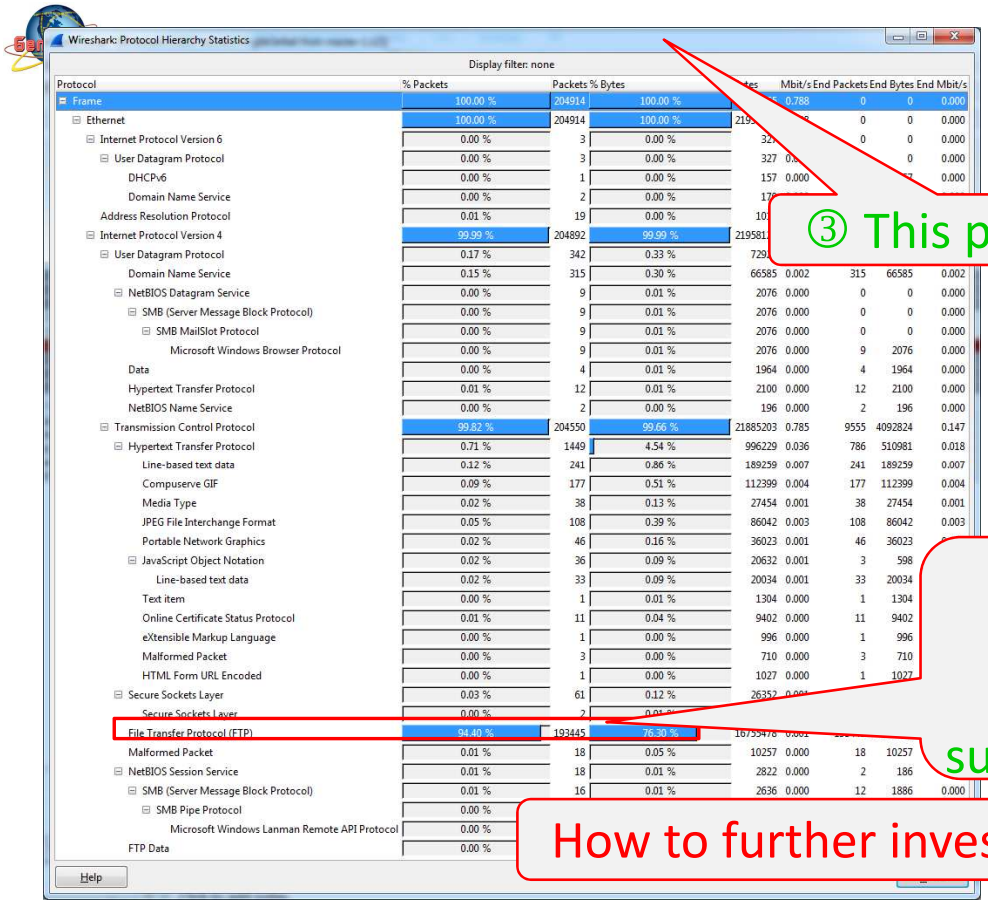

⑧ This window lists HTTP GET packets only
CaseStudy1.pcap
File Edit View Go
Filter: http.request.method == "GET"
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
10 1.063255 192.168.1.7 65.55.84.56 HTTP 516 GET /?ocid=iehp HTTP/1.1
65 1.471377 192.168.1.7 65.55.253.27 HTTP 1049 GET /c.gif?evt=impr&js=1&rid=45def397b817495bae983b0f282b2c51&exa=&pp=False&bd=&gnd=&cts=1
72 1.499904 192.168.1.7 207.46.193.176 HTTP 489 GET /action/MSN_Homepage_Remessaging_111808/nc?a=1 HTTP/1.1
77 1.508578 192.168.1.7 67.148.147.113 HTTP 527 GET /b?c1=2&c2=3000001&c7=http%3A%2F%2Fwww.msn.com%2F%3Focid%3Diehp&c9=&rn=1332688523104
84 1.538291 192.168.1.7 207.46.140.46 HTTP 802 GET /default.aspx?parsergroup=hops&fk=W&gp=P&optkey=default&cp=default&rf=&di=340&pi=7317&
91 1.593353 192.168.1.7 64.4.21.39 HTTP 680 GET /c.gif?udc=true&di=340&pi=7317&ps=95101&lng=en-us&tp=http%3A%2F%2Fwww.msn.com%2Fdefaul
94 1.598267 192.168.1.7 204.245.34.139 HTTP 578 GET /partner/primedns.gif HTTP/1.1
104 1.854517 192.168.1.7 204.245.34.139 HTTP 710 GET /sck?cn=_SS&r=http://www.msn.com/sck.aspx&form=MSN005&h=fe92cd76-4cb3-03b6-8afc-cd7f85
115 1.986251 192.168.1.7 63.235.36.105 HTTP 483 GET /qsonhs.aspx?form=MSN005&q= HTTP/1.1
119 2.043988 192.168.1.7 65.55.84.56 HTTP 837 GET /sck.aspx?cv=_SS%3dSID%3d7C73BA2D0DBD452E95CBDB3AFE1879B0%3b&h=fe92cd76-4cb3-03b6-8afc
122 2.056281 192.168.1.7 65.55.5.232 HTTP 636 GET /ADSAdClient31.dll?GetSAd=&DPJS=4&PN=MSFT&ID=15A78A4A61DA691C0F38896465DA6922&MUID=15A
129 2.188858 192.168.1.7 204.245.34.139 HTTP 636 GET /msnhomepagehistory.aspx?sid=7C73BA2D0DBD452E95CBDB3AFE1879B0&_=1332688523926 HTTP/1.1
149 2.414240 192.168.1.7 65.55.5.232 HTTP 638 GET /ADSAdClient31.dll?GetSAd=&DPJS=4&PN=MSFT&ID=15A78A4A61DA691C0F38896465DA6922&MUID=15A
164 2.502977 192.168.1.7 65.55.5.232 HTTP 638 GET /ADSAdClient31.dll?GetSAd=&DPJS=4&PN=MSFT&ID=15A78A4A61DA691C0F38896465DA6922&MUID=15A
165 2.503215 192.168.1.7 65.55.5.232 HTTP 638 GET /ADSAdClient31.dll?GetSAd=&DPJS=4&PN=MSFT&ID=15A78A4A61DA691C0F38896465DA6922&MUID=15A
169 2.533468 192.168.1.7 65.55.5.232 HTTP 638 GET /ADSAdClient31.dll?GetSAd=&DPJS=4&PN=MSFT&ID=15A78A4A61DA691C0F38896465DA6922&MUID=15A
183 2.651686 192.168.1.7 75.98.29.8 HTTP 678 GET /cookE/geoip/iframe?spacedesc=2186184_1116771_300x120_2186181_2186184&target=_blank&&C
199 2.875084 192.168.1.7 75.98.29.8 HTTP 851 GET /xl/PROD/19706/creatives/mat_boygrl_fbsingle_RePopScrollImgDsslveFadeForm_vsgeo_Mgendc
Frame 10: 516 bytes on wire (4128 bits), 516 bytes captured
Ethernet II, Src: vmware_db:f6:71 (00:0c:29:db:f6:71), Dst
Internet Protocol Version 4, Src: 192.168.1.7 (192.168.1.
Transmission Control Protocol, Src Port: 49635 (49635),
80 (80), Seq: 1, Ack: 1, Len: 462
Hypertext Transfer Protocol
This pane has fewer packets, easier to study
File: "C:\tmp\Boot-capture\CaseStudy1.pca... Packets: 266 · Displayed: 22 (8.3%) · Load time: 0:00.005
Profile: Default

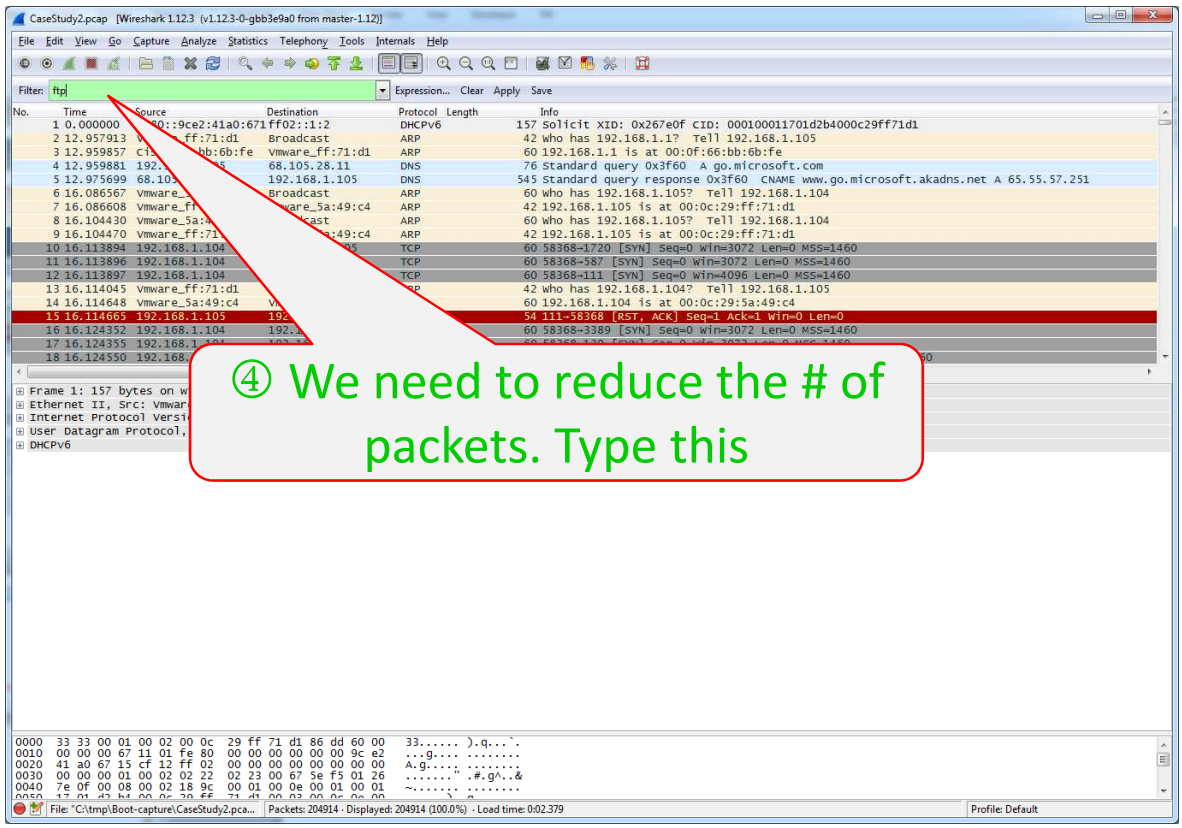
Exercise ❷

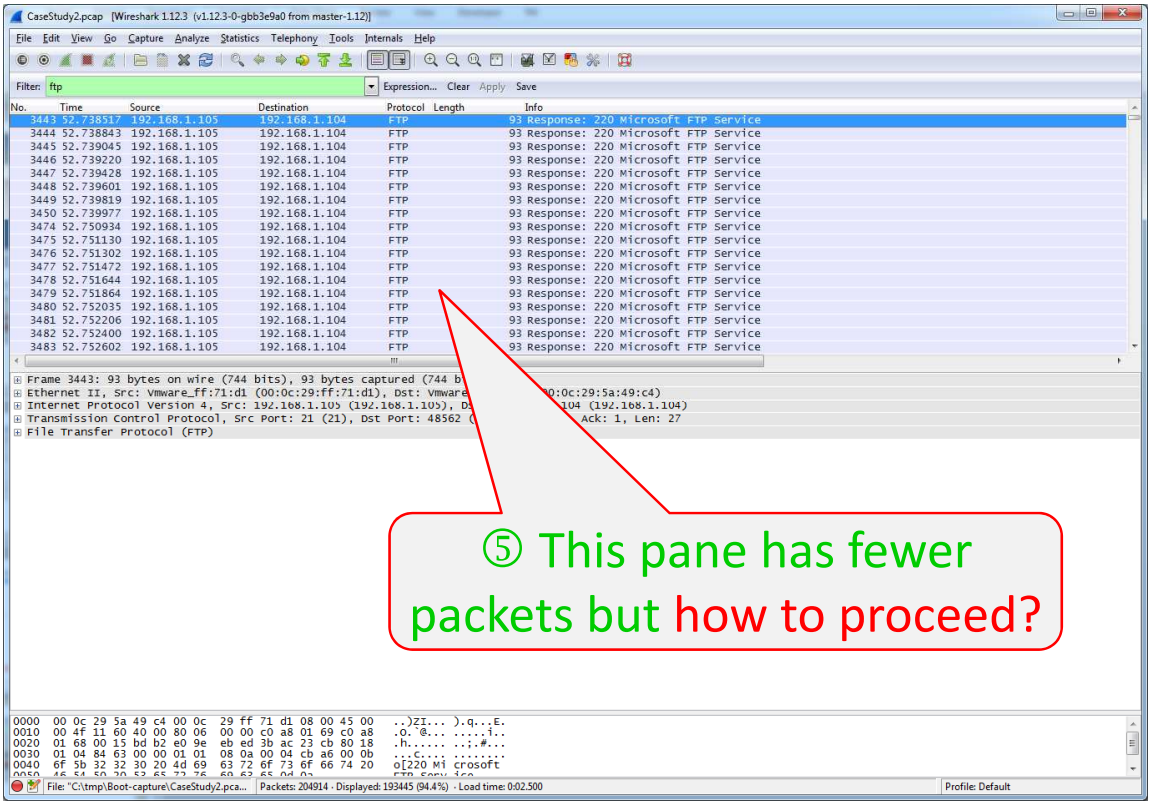

Wireshark
SimpleCapt...
CaseStudy...
CaseStudy...
WebCaptur...
Recycle Bin
Start
Double click on this
This example has more advanced features (find attacks from packets!)

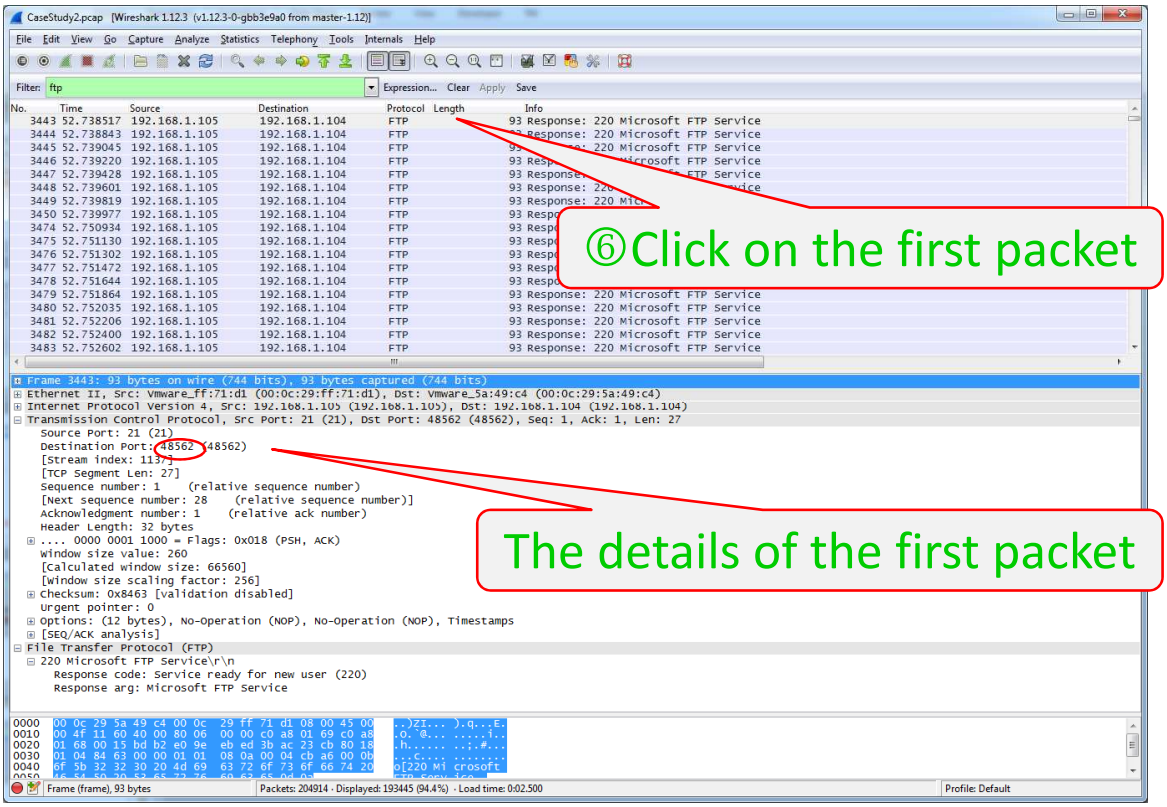

CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
No. Time Source Destination Protocol Length Info
1 0.000000 fe80::9ce2:41a0:671 ff02::1:2 DHCPv6 157 Solicit XID: 0x267e0f CID: 000100011701d2b4000c29ff71d1
2 12.957913 Vmware_ff:71:d1 Broadcast ARP 42 Who has 192.168.1.1? Tell 192.168.1.105
3 12.959857 Cisco-Li_bb:6b:fe Vmware_ff:71:d1 ARP 60 192.168.1.1 is at 00:0f:66:bb:6b:fe
4 12.959881 192.168.1.105 68.105.28.11 DNS 76 Standard query 0x3f60 A go.microsoft.com
5 12.975699 68.105.28.11 192.168.1.105 DNS 545 Standard query response 0x3f60 CNAME www.go.microsoft.akadns.net A 65.55.57.251
6 16.086567 Vmware_5a:49:c4 Broadcast ARP 60 Who has 192.168.1.105? Tell 192.168.1.104
7 16.086608 Vmware_ff:71:d1 Vmware_5a:49:c4 ARP 42 192.168.1.105 is at 00:0c:29:ff:71:d1
8 16.104430 Vmware_5a:49:c4 Broadcast ARP 60 Who has 192.168.1.105? Tell 192.168.1.104
9 16.104470 Vmware_ff:71:d1 Vmware_5a:49:c4 ARP 42 192.168.1.105 is at 00:0c:29:ff:71:d1
10 16.113894 192.168.1.104 192.168.1.105 TCP 60 58368→1720 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
11 16.113896 192.168.1.104 192.168.1.105 TCP 60 58368→587 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
12 16.113897 192.168.1.104 192.168.1.105 TCP 60 58368→111 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
13 16.114045 Vmware_ff:71:d1 Broadcast ARP 42 Who has 192.168.1.104? Tell 192.168.1.105
14 16.114648 Vmware_5a:49:c4 Vmware_ff:71:d1 ARP 60 192.168.1.104 is at 00:0c:29:5a:49:c4
15 16.114665 192.168.1.105 192.168.1.104 TCP 54 111→58368 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16 16.124352 192.168.1.104 192.168.1.105 TCP 60 58368→3389 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
17 16.124355 192.168.1.104 192.168.1.105 TCP 60 58368→139 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
18 16.124550 192.168.1.105 192.168.1.104 TCP 58 3389→58368 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
Frame 1: 157 bytes on wire (1256 bits)
Ethernet II, Src: Vmware_ff:71
Internet Protocol Version 6,
User Datagram Protocol, Src
DHCPv6
0000 33 33 00 01 00 02 00 0c 29 ff 71 d1 86 dd 60 00 33.....).q...`.
0010 00 00 00 67 11 01 fe 80 00 00 00 00 00 00 9c e2 ...g.... ....
0020 41 a0 67 15 cf 12 ff 02 00 00 00 00 00 00 00 00 A.g..... ....
0030 00 00 00 01 00 02 02 22 02 23 00 67 5e f5 01 26 ....#.g^..&
0040 7e 0f 00 08 00 02 18 9c 00 01 00 0e 00 01 00 01 ~..... ....
File: "C:\tmp\Boot-capture\CaseStudy2.pca... Packets: 204914 · Displayed: 204914 (100.0%) · Load time: 0:02.379
Profile: Default
This example has many (too many?) packets
Are there any attack packets here?

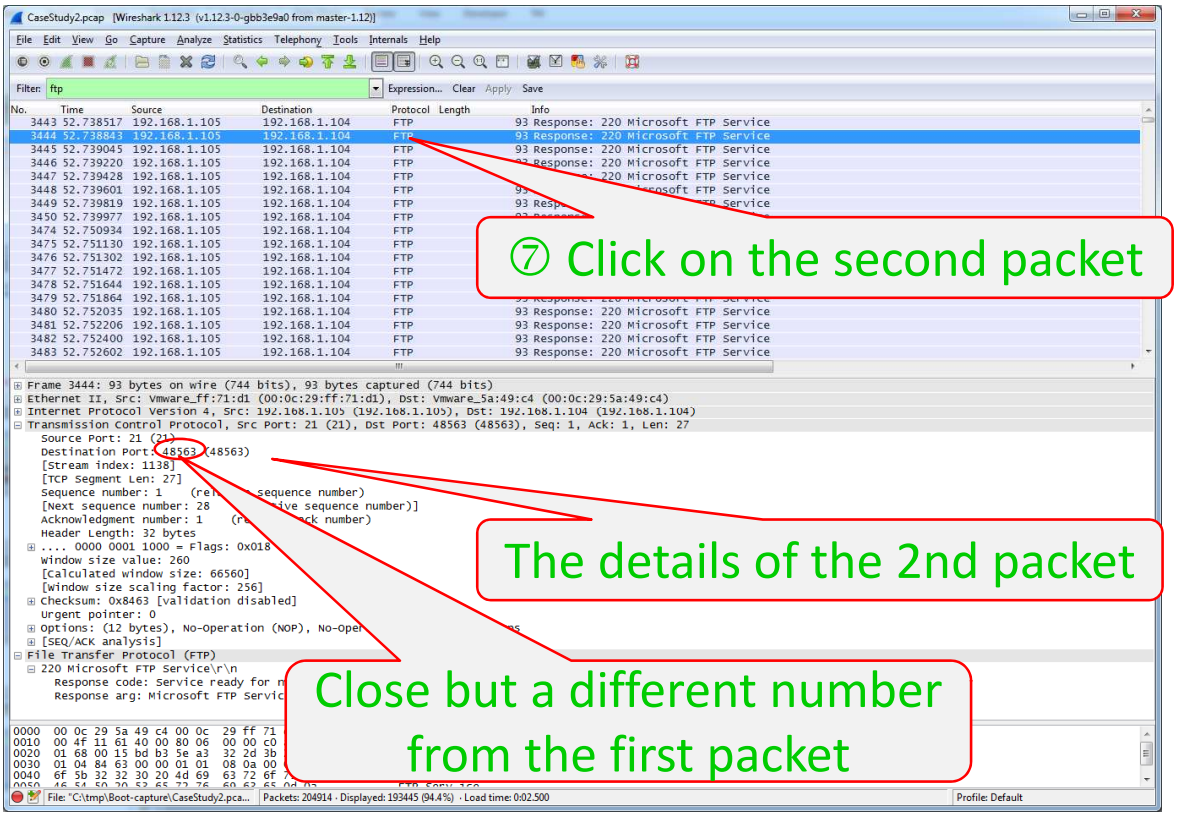

① Select Statistics
② Select
Protocol Hierarchy

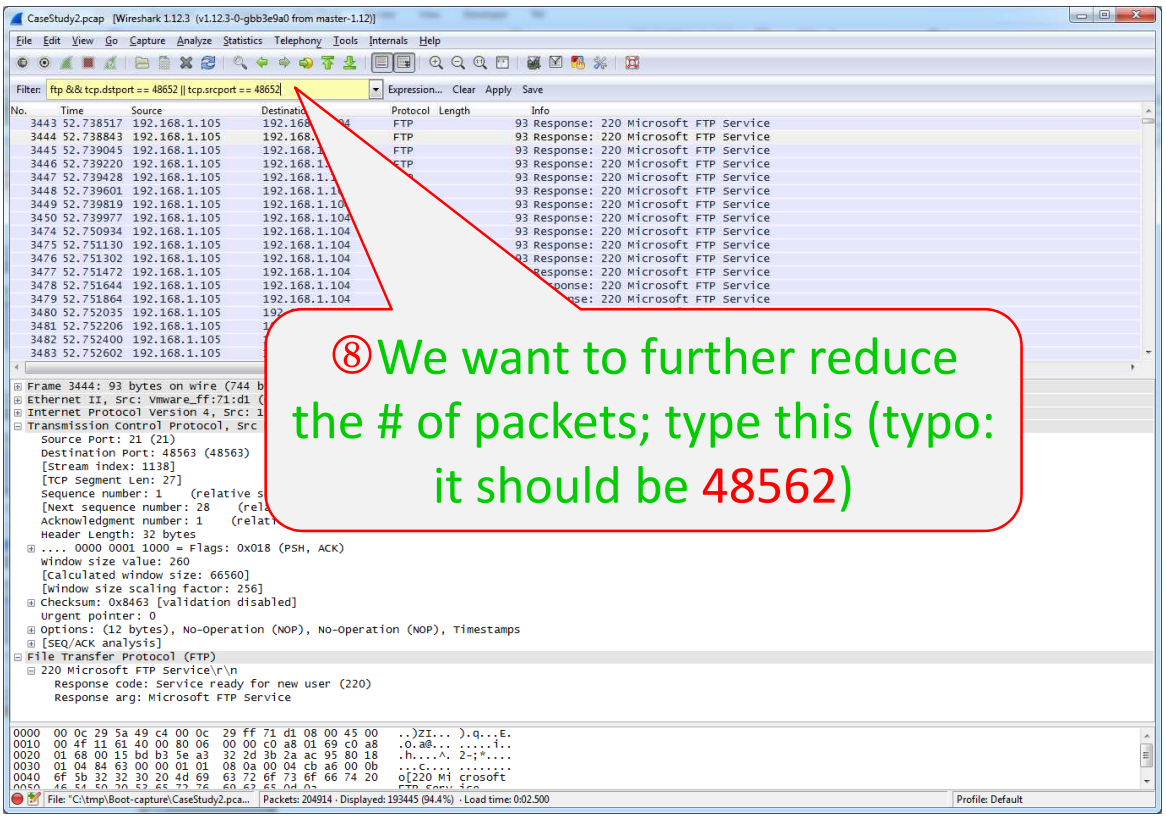

Wireshark: Protocol Hierarchy Statistics
Display filter: none
③ This pops up
94%?
This is suspicious
How to further investigate?

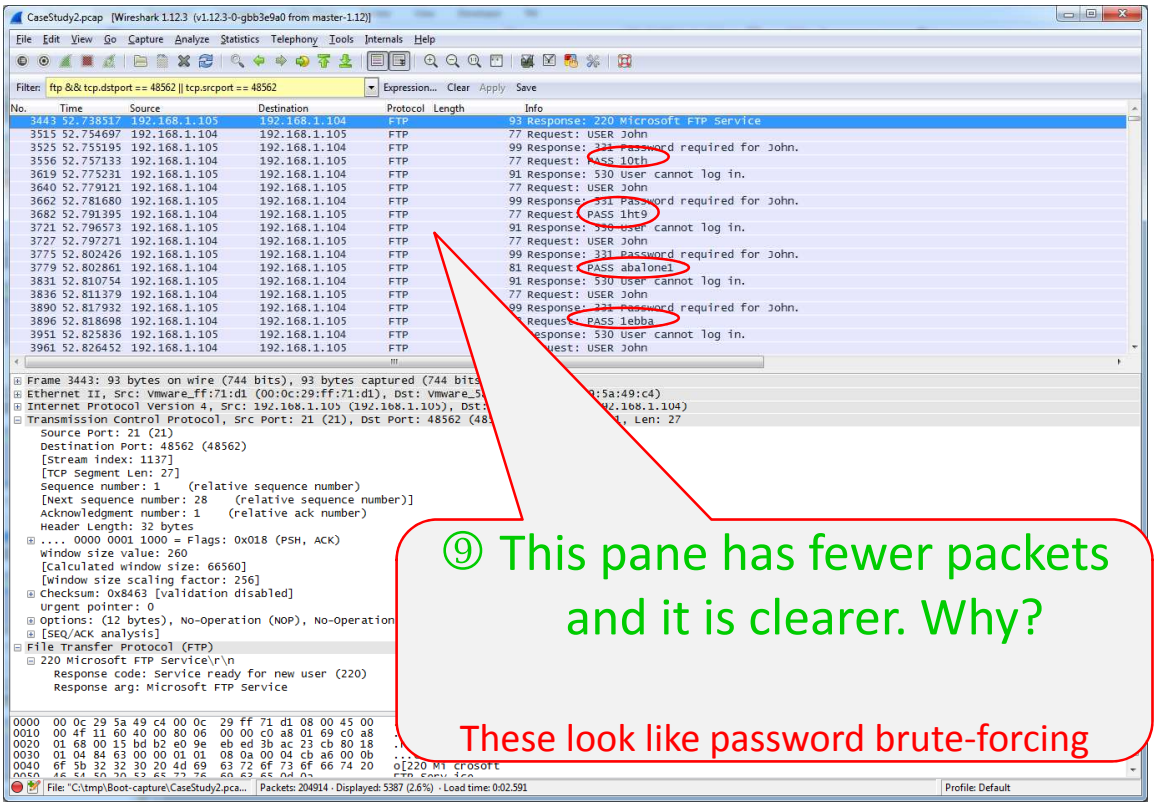

④ We need to reduce the # of packets. Type this

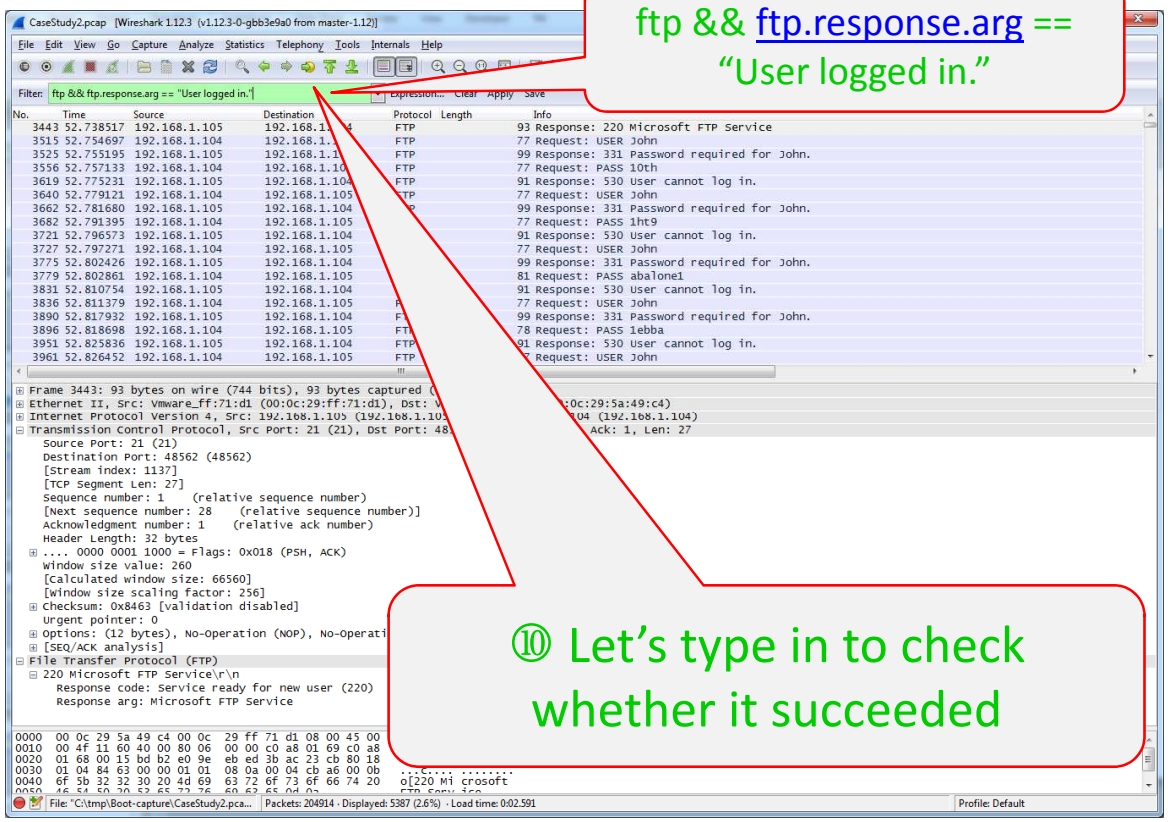

⑤ This pane has fewer packets but how to proceed?

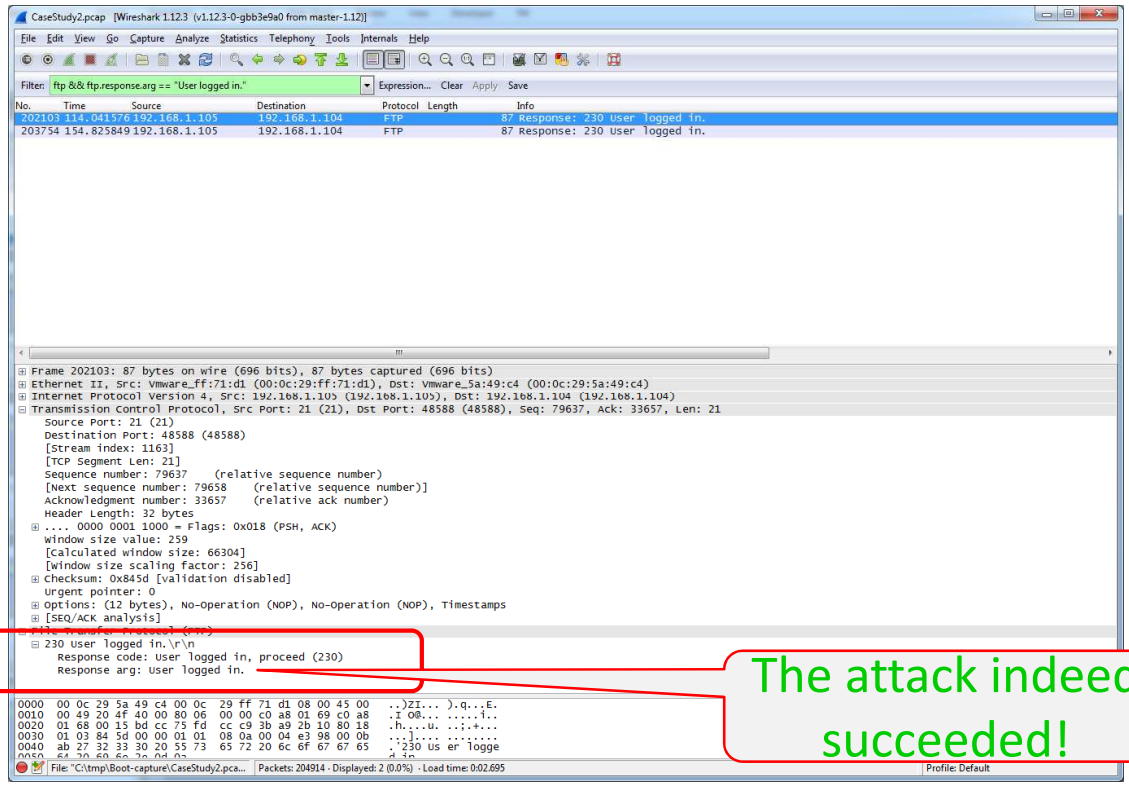

⑥Click on the first packet
The details of the first packet

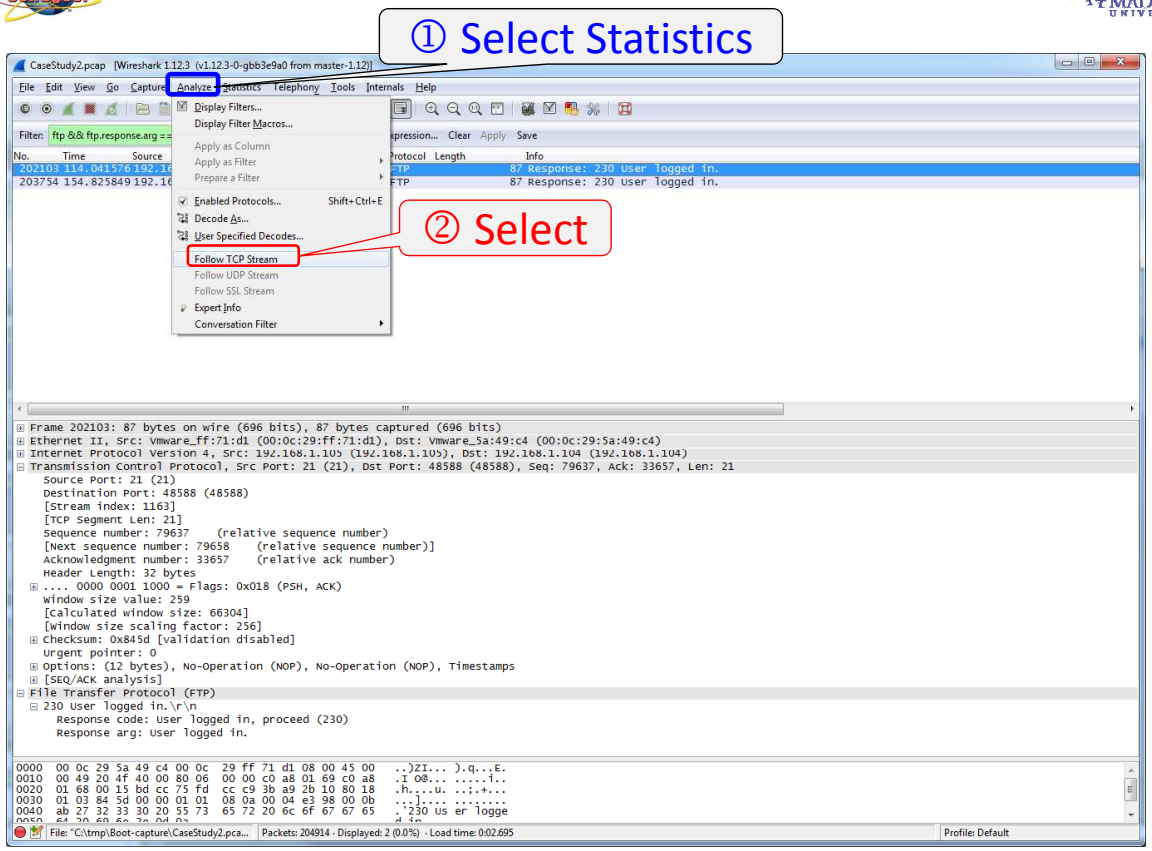

⑦ Click on the second packet
The details of the 2nd packet
Close but a different number from the first packet

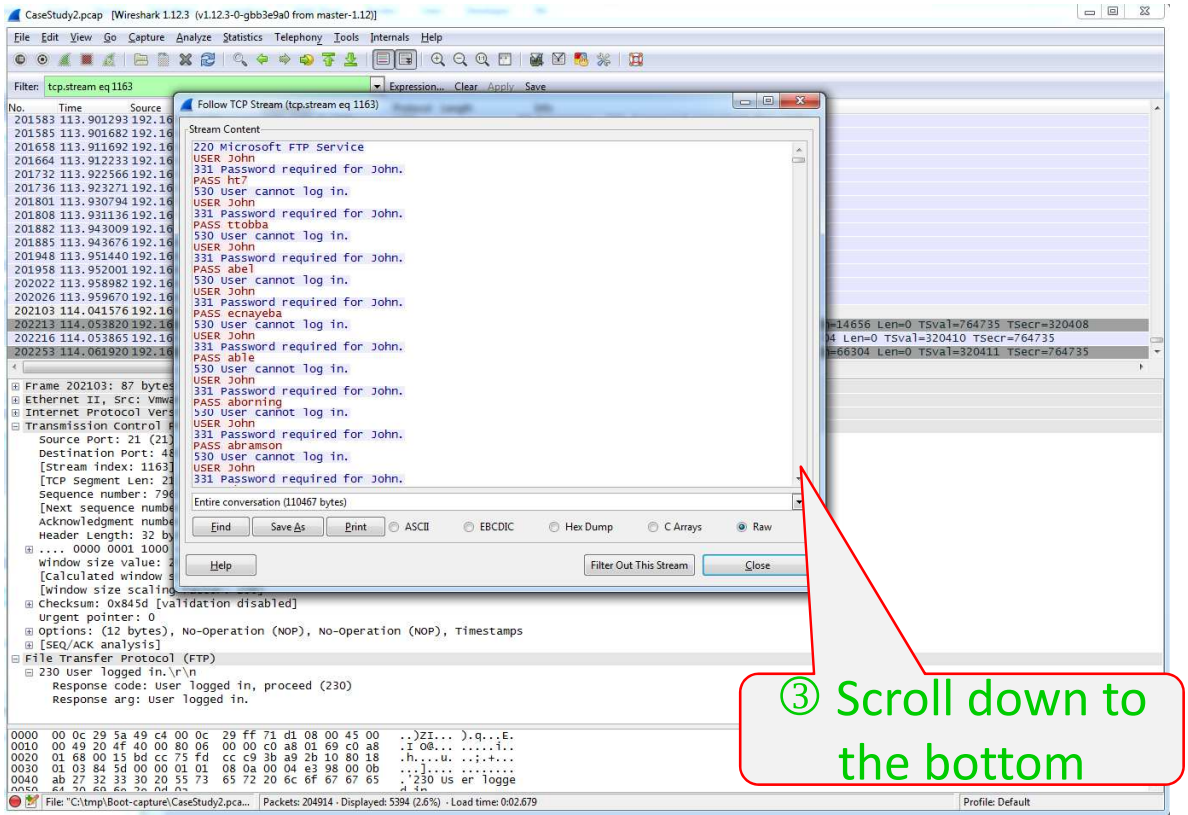

CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Filter: ftp && tcp.dstport == 48652 || tcp.srcport == 48652
⑧We want to further reduce the # of packets; type this (typo: it should be 48562)

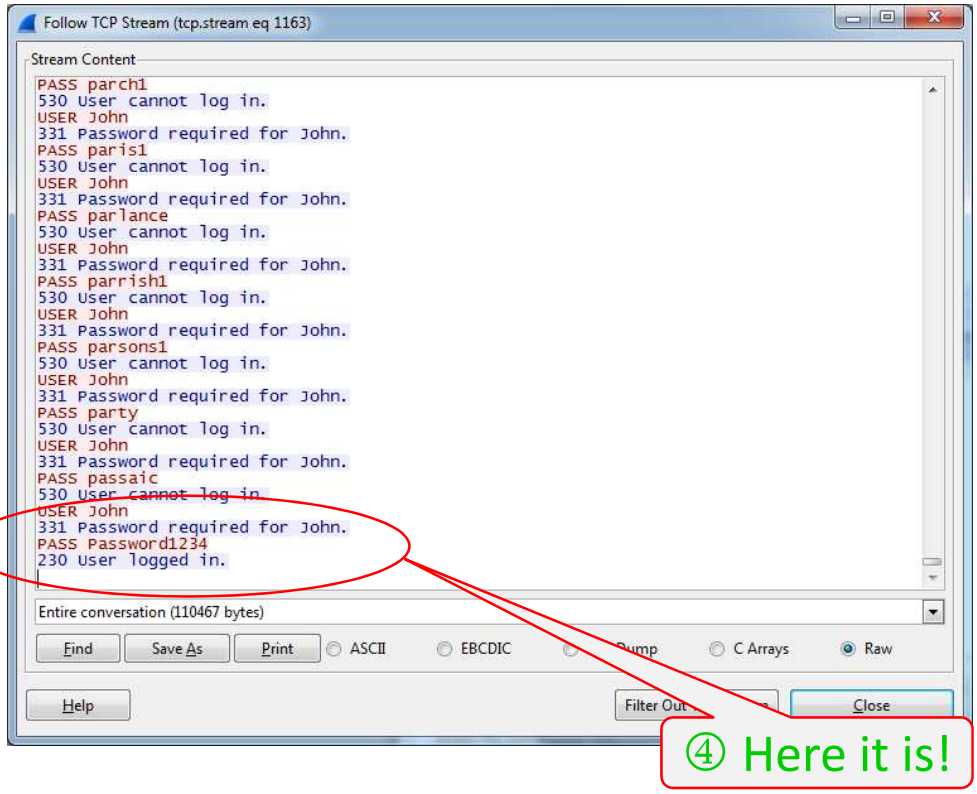

CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Filter: ftp && tcp.dstport == 48562 || tcp.srcport == 48562
⑨ This pane has fewer packets and it is clearer. Why?
These look like password brute-forcing

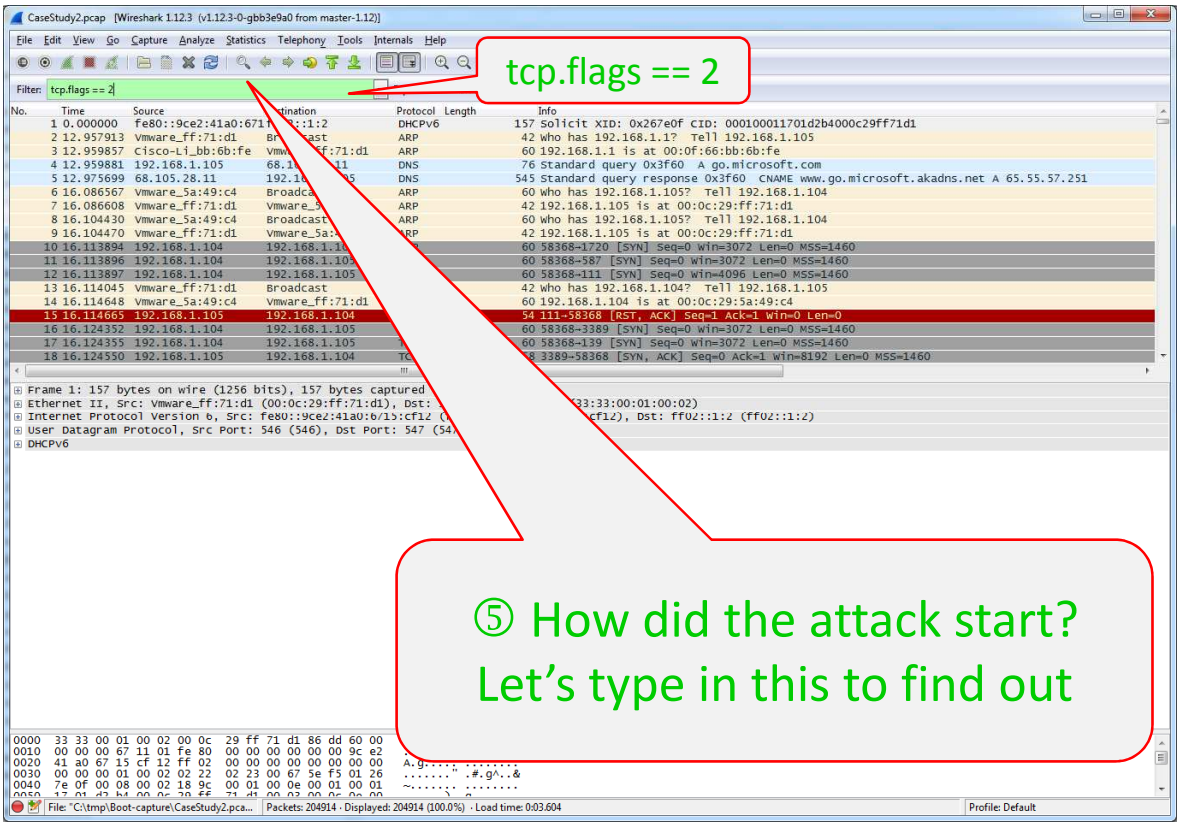

ftp && ftp.response.arg == "User logged in."
⑩ Let's type in to check whether it succeeded

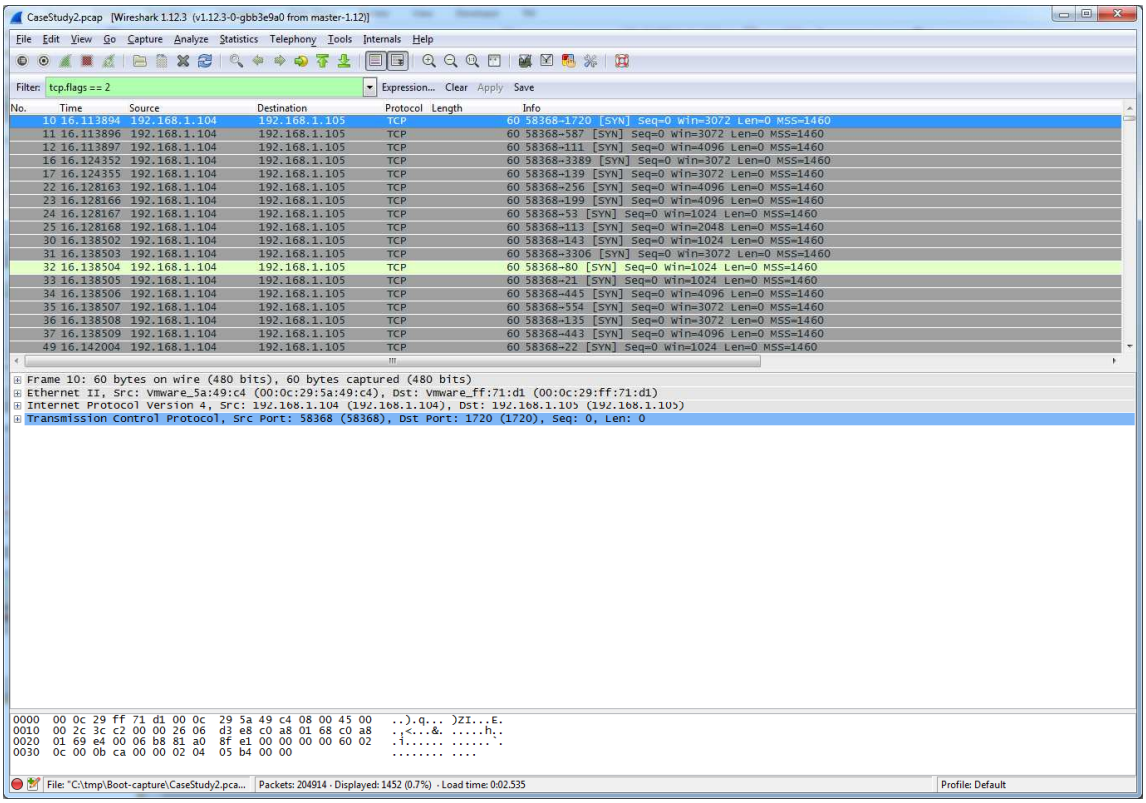

The attack indeed succeeded!

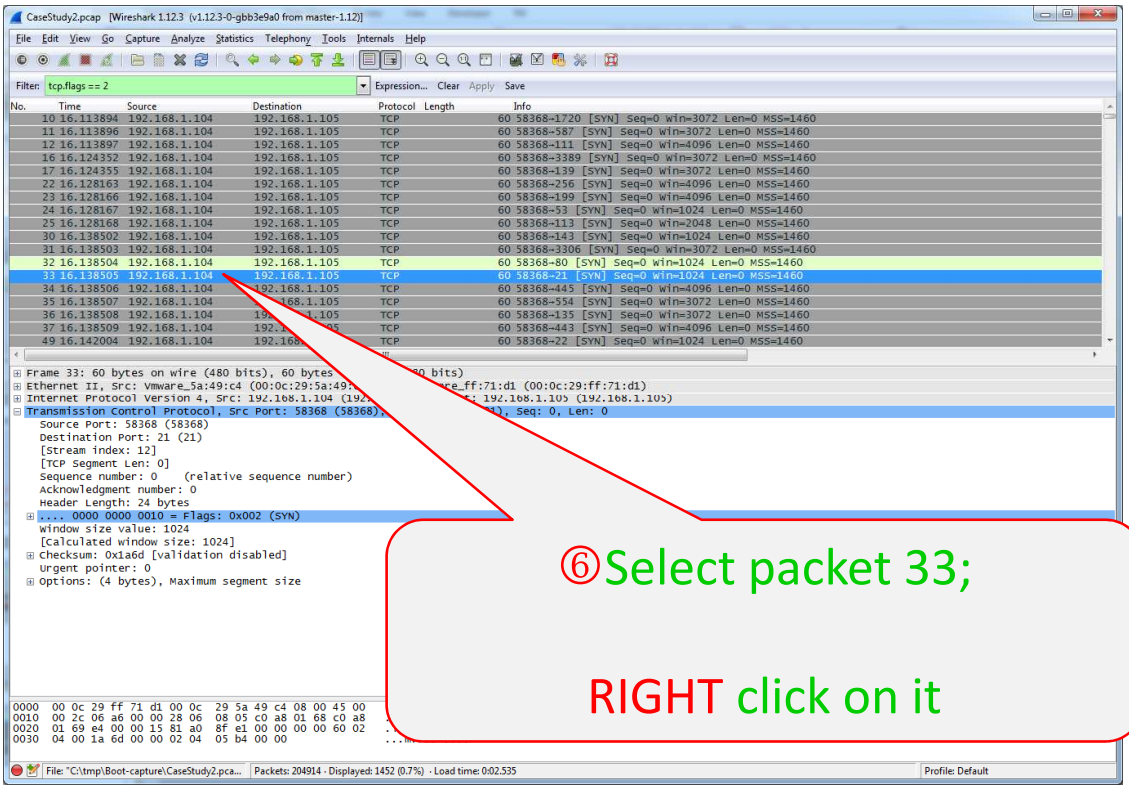

① Select Statistics
② Select
Follow TCP Stream

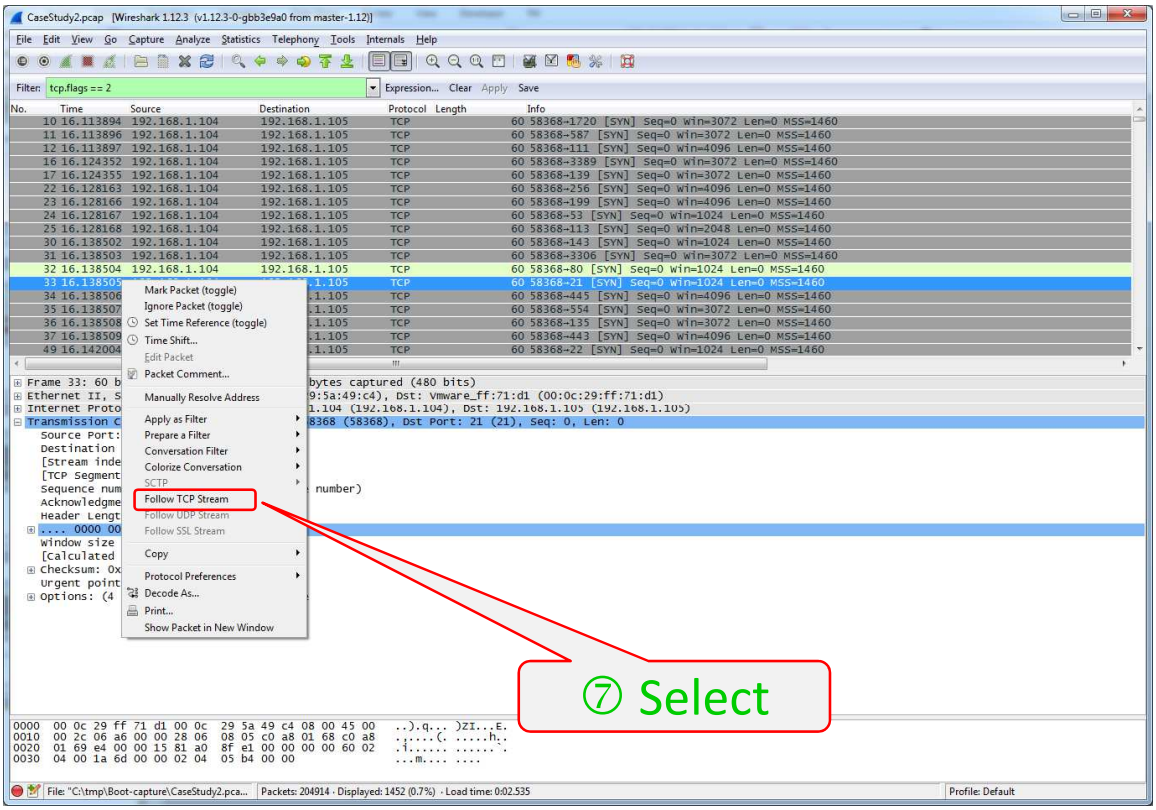

Follow TCP Stream (tcp.stream eq 1163)
Stream Content
220 Microsoft FTP Service
USER John
331 Password required for John.
PASS ht7
530 User cannot log in.
USER John
331 Password required for John.
PASS ttobba
530 User cannot log in.
USER John
331 Password required for John.
PASS abel
530 User cannot log in.
USER John
331 Password required for John.
PASS ecnayeba
530 User cannot log in.
USER John
331 Password required for John.
PASS able
530 User cannot log in.
USER John
331 Password required for John.
PASS aborning
530 User cannot log in.
USER John
331 Password required for John.
PASS abramson
530 User cannot log in.
USER John
331 Password required for John.
Entire conversation (110467 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
Raw
Help
Filter Out This Stream
Close
③ Scroll down to the bottom

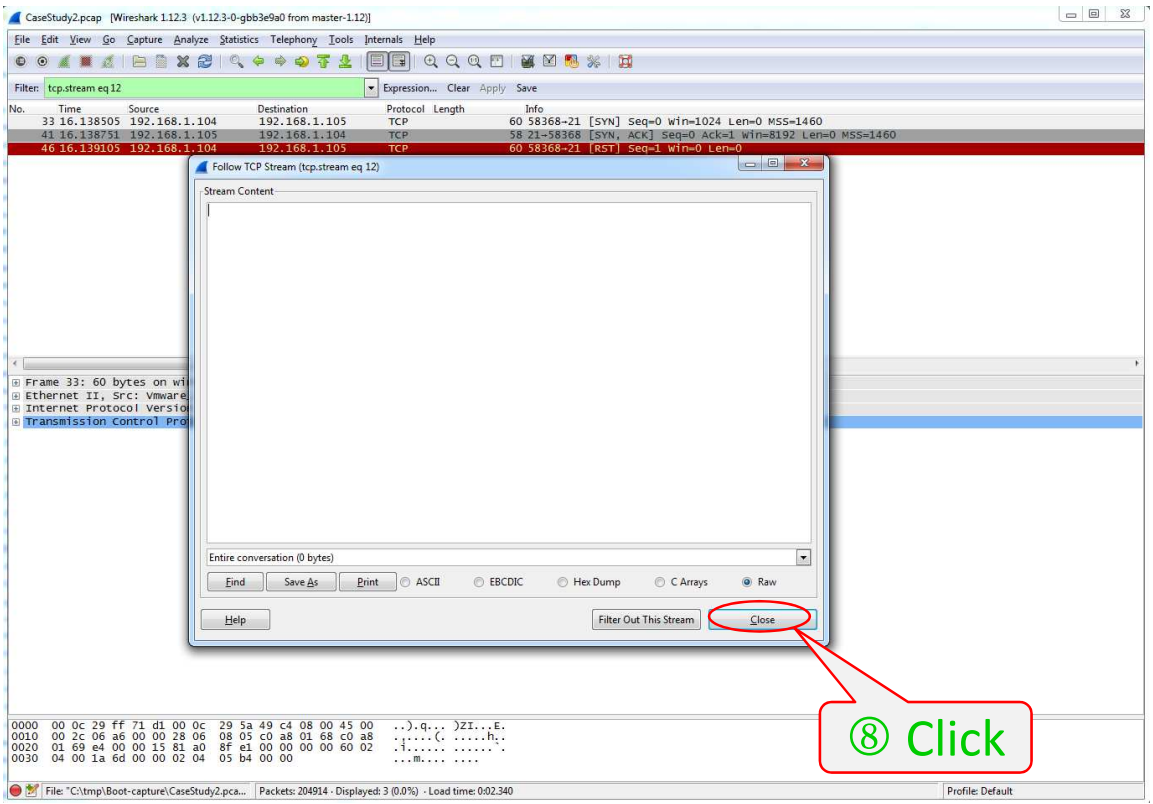

Follow TCP Stream (tcp.stream eq 1163)
Stream Content
PASS parch1
530 User cannot log in.
USER John
331 Password required for John.
PASS paris1
530 User cannot log in.
USER John
331 Password required for John.
PASS parlance
530 User cannot log in.
USER John
331 Password required for John.
PASS parrish1
530 User cannot log in.
USER John
331 Password required for John.
PASS parsons1
530 User cannot log in.
USER John
331 Password required for John.
PASS party
530 User cannot log in.
USER John
331 Password required for John.
PASS passaic
530 User cannot log in.
USER John
331 Password required for John.
PASS Password1234
230 User logged in.
Entire conversation (110467 bytes)
Find
Save As
Print
ASCII
EBCDIC
C Arrays
Raw
Help
Filter Out
Close
④ Here it is!

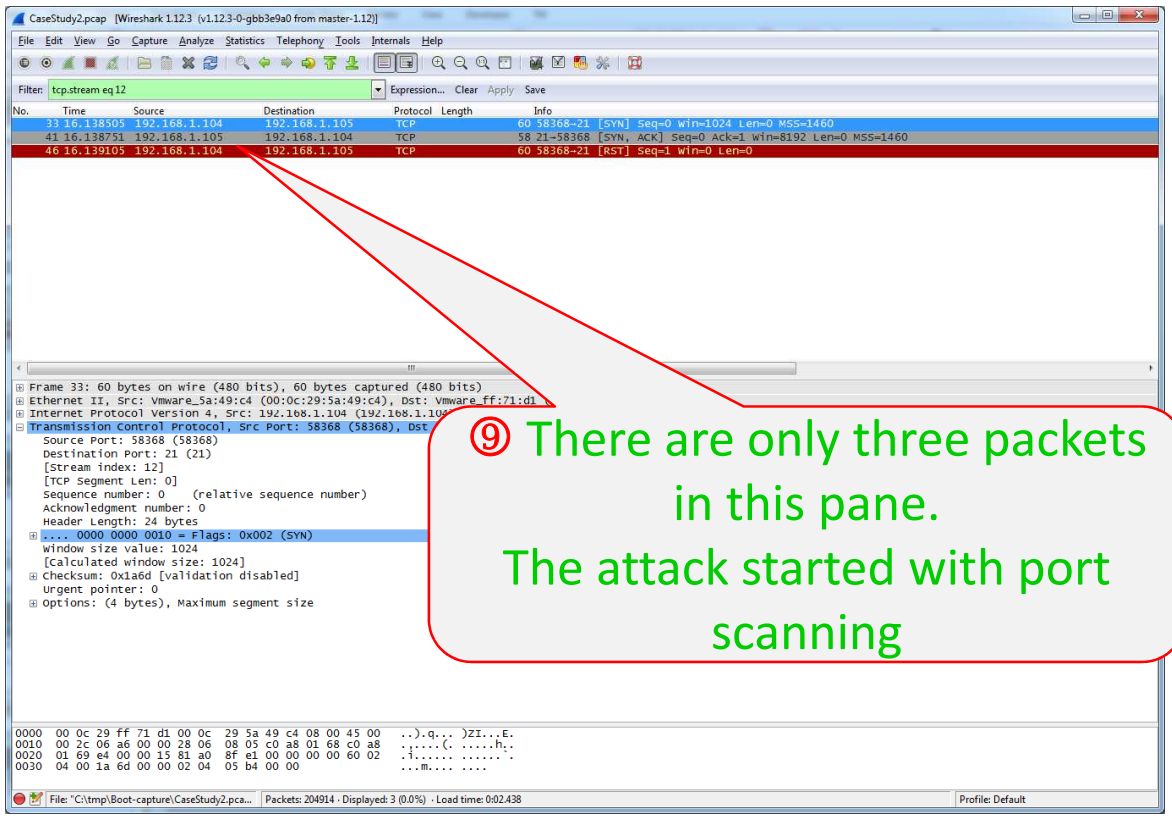

CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.flags == 2
tcp.flags == 2
No. Time Source Destination Protocol Length Info
1 0.000000 fe80::9ce2:41a0:671f 1:2 DHCPv6 157 Solicit XID: 0x267e0f CID: 000100011701d2b4000c29ff71d1
2 12.957913 Vmware_ff:71:d1 Broadcast ARP 42 Who has 192.168.1.1? Tell 192.168.1.105
3 12.959857 Cisco-Li_bb:6b:fe Vmware_ff:71:d1 ARP 60 192.168.1.1 is at 00:0f:66:bb:6b:fe
4 12.959881 192.168.1.105 68.1 11 DNS 76 Standard query 0x3f60 A go.microsoft.com
5 12.975699 68.105.28.11 192.1 5 DNS 545 Standard query response 0x3f60 CNAME www.go.microsoft.akadns.net A 65.55.57.251
6 16.086567 Vmware_5a:49:c4 Broadca ARP 60 Who has 192.168.1.105? Tell 192.168.1.104
7 16.086608 Vmware_ff:71:d1 Vmware_5 ARP 42 192.168.1.105 is at 00:0c:29:ff:71:d1
8 16.104430 Vmware_5a:49:c4 Broadcast ARP 60 Who has 192.168.1.105? Tell 192.168.1.104
9 16.104470 Vmware_ff:71:d1 Vmware_5a: ARP 42 192.168.1.105 is at 00:0c:29:ff:71:d1
10 16.113894 192.168.1.104 192.168.1.1 60 58368→1720 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
11 16.113896 192.168.1.104 192.168.1.105 60 58368→587 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
12 16.113897 192.168.1.104 192.168.1.105 60 58368→111 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
13 16.114045 Vmware_ff:71:d1 Broadcast 42 Who has 192.168.1.104? Tell 192.168.1.105
14 16.114648 Vmware_5a:49:c4 Vmware_ff:71:d1 60 192.168.1.104 is at 00:0c:29:5a:49:c4
15 16.114665 192.168.1.105 192.168.1.104 54 111→58368 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16 16.124352 192.168.1.104 192.168.1.105 60 58368→3389 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
17 16.124355 192.168.1.104 192.168.1.105 60 58368→139 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
18 16.124550 192.168.1.105 192.168.1.104 8 3389→58368 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
Frame 1: 157 bytes on wire (1256 bits), 157 bytes captured
Ethernet II, Src: Vmware_ff:71:d1 (00:0c:29:ff:71:d1), Dst: 33:33:00:01:00:02)
Internet Protocol Version 6, Src: fe80::9ce2:41a0:671:cf12 cf12), Dst: ff02::1:2 (ff02::1:2)
User Datagram Protocol, Src Port: 546 (546), Dst Port: 547 (54
DHCPv6
0000 33 33 00 01 00 02 00 0c 29 ff 71 d1 86 dd 60 00
0010 00 00 00 67 11 01 fe 80 00 00 00 00 00 00 9c e2
0020 41 a0 67 15 cf 12 ff 02 00 00 00 00 00 00 00 00
0030 00 00 00 01 00 02 02 22 02 23 00 67 5e f5 01 26
0040 7e 0f 00 08 00 02 18 9c 00 01 00 0e 00 01 00 01
File: "C:\tmp\Boot-capture\CaseStudy2.pca... Packets: 204914 · Displayed: 204914 (100.0%) · Load time: 0:03.604
Profile: Default
⑤ How did the attack start?
Let's type in this to find out


CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.flags == 2
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
10 16.113894 192.168.1.104 192.168.1.105 TCP 60 58368→1720 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
11 16.113896 192.168.1.104 192.168.1.105 TCP 60 58368→587 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
12 16.113897 192.168.1.104 192.168.1.105 TCP 60 58368→111 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
16 16.124352 192.168.1.104 192.168.1.105 TCP 60 58368→3389 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
17 16.124355 192.168.1.104 192.168.1.105 TCP 60 58368→139 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
22 16.128163 192.168.1.104 192.168.1.105 TCP 60 58368→256 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
23 16.128166 192.168.1.104 192.168.1.105 TCP 60 58368→199 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
24 16.128167 192.168.1.104 192.168.1.105 TCP 60 58368→53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
25 16.128168 192.168.1.104 192.168.1.105 TCP 60 58368→113 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
30 16.138502 192.168.1.104 192.168.1.105 TCP 60 58368→143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
31 16.138503 192.168.1.104 192.168.1.105 TCP 60 58368→3306 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
32 16.138504 192.168.1.104 192.168.1.105 TCP 60 58368→80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
33 16.138505 192.168.1.104 192.168.1.105 TCP 60 58368→21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
34 16.138506 192.168.1.104 192.168.1.105 TCP 60 58368→445 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
35 16.138507 192.168.1.104 192.168.1.105 TCP 60 58368→554 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
36 16.138508 192.168.1.104 192.168.1.105 TCP 60 58368→135 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
37 16.138509 192.168.1.104 192.168.1.105 TCP 60 58368→443 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
49 16.142004 192.168.1.104 192.168.1.105 TCP 60 58368→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: Vmware_5a:49:c4 (00:0c:29:5a:49:c4), Dst: Vmware_ff:71:d1 (00:0c:29:ff:71:d1)
Internet Protocol Version 4, Src: 192.168.1.104 (192.168.1.104), Dst: 192.168.1.105 (192.168.1.105)
Transmission Control Protocol, Src Port: 58368 (58368), Dst Port: 1720 (1720), Seq: 0, Len: 0
0000 00 0c 29 ff 71 d1 00 0c 29 5a 49 c4 08 00 45 00 ..).q...)ZI...E.
0010 00 2c 3c c2 00 00 26 06 d3 e8 c0 a8 01 68 c0 a8 .,<...&. .....h..
0020 01 69 e4 00 06 b8 81 a0 8f e1 00 00 00 00 60 02 .i.....
0030 0c 00 0b ca 00 00 02 04 05 b4 00 00 ....
File: "C:\tmp\Boot-capture\CaseStudy2.pca... Packets: 204914 · Displayed: 1452 (0.7%) · Load time: 0:02.535
Profile: Default


CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.flags == 2
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
10 16.113894 192.168.1.104 192.168.1.105 TCP 60 58368→1720 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
11 16.113896 192.168.1.104 192.168.1.105 TCP 60 58368→587 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
12 16.113897 192.168.1.104 192.168.1.105 TCP 60 58368→111 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
16 16.124352 192.168.1.104 192.168.1.105 TCP 60 58368→3389 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
17 16.124355 192.168.1.104 192.168.1.105 TCP 60 58368→139 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
22 16.128163 192.168.1.104 192.168.1.105 TCP 60 58368→256 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
23 16.128166 192.168.1.104 192.168.1.105 TCP 60 58368→199 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
24 16.128167 192.168.1.104 192.168.1.105 TCP 60 58368→53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
25 16.128168 192.168.1.104 192.168.1.105 TCP 60 58368→113 [SYN] Seq=0 Win=2048 Len=0 MSS=1460
30 16.138502 192.168.1.104 192.168.1.105 TCP 60 58368→143 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
31 16.138503 192.168.1.104 192.168.1.105 TCP 60 58368→3306 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
32 16.138504 192.168.1.104 192.168.1.105 TCP 60 58368→80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
33 16.138505 192.168.1.104 192.168.1.105 TCP 60 58368→21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
34 16.138506 192.168.1.104 192.168.1.105 TCP 60 58368→445 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
35 16.138507 192.168.1.104 192.168.1.105 TCP 60 58368→554 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
36 16.138508 192.168.1.104 192.168.1.105 TCP 60 58368→135 [SYN] Seq=0 Win=3072 Len=0 MSS=1460
37 16.138509 192.168.1.104 192.168.1.105 TCP 60 58368→443 [SYN] Seq=0 Win=4096 Len=0 MSS=1460
49 16.142004 192.168.1.104 192.168.1.105 TCP 60 58368→22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
Source Port: 58368 (58368)
Destination Port: 21 (21)
[Stream index: 12]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 0
Header Length: 24 bytes
.... 0000 0000 0010 = Flags: 0x002 (SYN)
Window size value: 1024
[Calculated window size: 1024]
Checksum: 0x1a6d [validation disabled]
Urgent pointer: 0
Options: (4 bytes), Maximum segment size
⑥Select packet 33;
RIGHT click on it


CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Filter: tcp.flags == 2
Mark Packet (toggle)
Ignore Packet (toggle)
Set Time Reference (toggle)
Time Shift...
Edit Packet
Packet Comment...
Manually Resolve Address
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow TCP Stream
Follow UDP Stream
Follow SSL Stream
Copy
Protocol Preferences
Decode As...
Print...
Show Packet in New Window
⑦ Select


CaseStudy2.pcap [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
Filter: tcp.stream eq 12
Follow TCP Stream (tcp.stream eq 12)
Stream Content
Entire conversation (0 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
Raw
Help
Filter Out This Stream
Close
⑧ Click



Summary

- Prerequisite: network packet & packet analyzer: (header, data)
  - Enveloped letters inside another envelope
- Exercises
  - ❶Basic network traffic analysis
    - SimpleCapture.pcap, WebCapture.pcap
  - ❷Gather information and statistics
    - CaseStudy1.pcap, CaseStudy2.pcap
    - Traffic searches: protocol hierarchy, HTTP requests, conversations, filters; attack analysis

Notes, with the same content,
are included

Network Sniffing and Packet Analysis Exercise

What is packet analysis and how to capture network traffic?

A **packet analyzer** is a piece of computer hardware or software that can intercept and log traffic passing over a digital network. When a network request is made (i.e. a web-page search, sent email message…) information is sent across the network from the source location to the destination via multiple data streams/packets. These streams contain header information that describes among other things the source of the request, the destination of the request, the type of data contained in the packet, various information describing the transaction and is then followed by the actual data. On simple web search can generate many data packets.

In this exercise we will analyze some previously captured traffic and explain the contents of the data in detail. We will discuss how to filter captured data streams to limit simplify and fine tune analysis. Through these demonstrations we will enlighten you on safety procedures and risks involved in running certain applications. Specifically we will analyze the following types of network traffic:

- Web Server – http
- File transfer Protocol – FTP
- Port scan
- Password cracking attempt

For these exercises we will use the Wireshark software application that has been installed on your virtual machine. Wireshark, formerly known as Ethereal, is a very powerful tool for network analysis. Wireshark is especially popular because it runs on Windows, Mac OS and Linux. It is a network packet analyzer that can peer inside the network and examine the details of traffic at varying levels. The information it can show you range from application-level information to the actual bits in a single packet.

We will analyze network traffic that was previously captured and has also been installed on your machine. You will need to connect to the vSphere Web Client. Start up and log in to the Snapshot entitled Packet Sniffing Exercise.

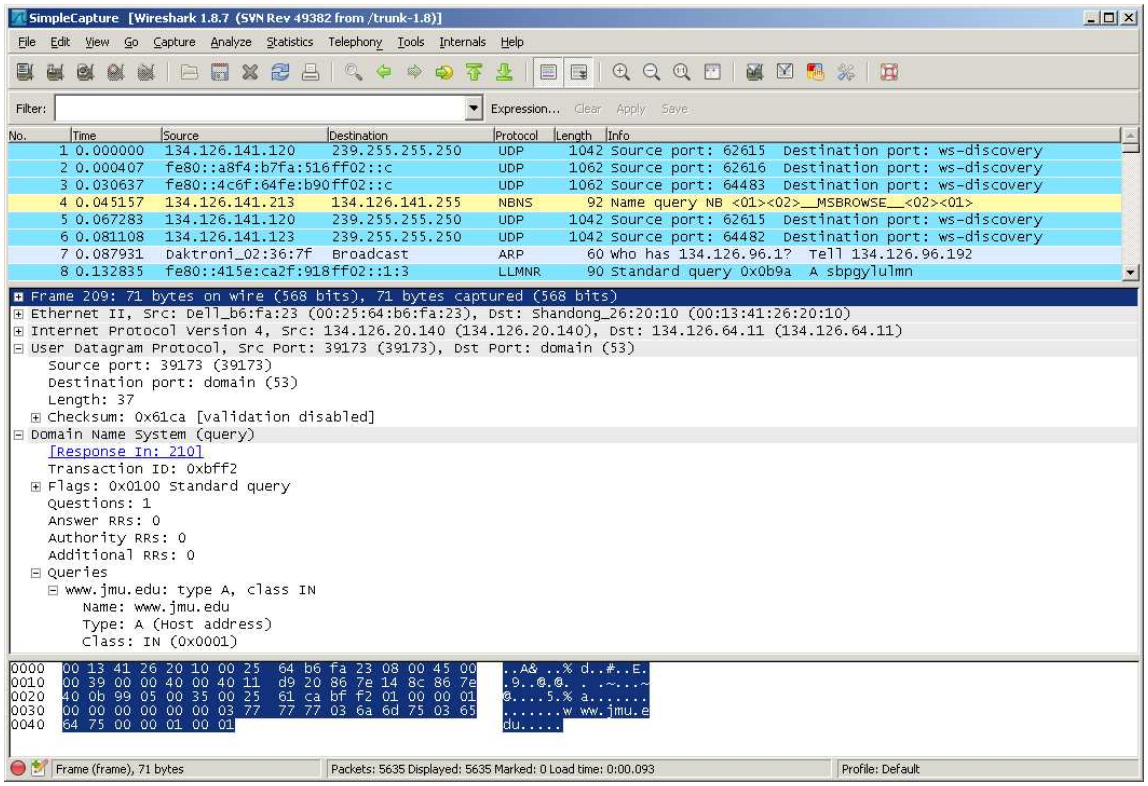
Exercise 1 –Using Wireshark to analyze basic network traffic

We will begin by analyzing a simple network traffic capture. *Double-click* on the ***Wireshark*** desktop icon. When Wireshark is used to capture and save network traffic it is saved in format known as a ***.pcap*** (packet capture) file.

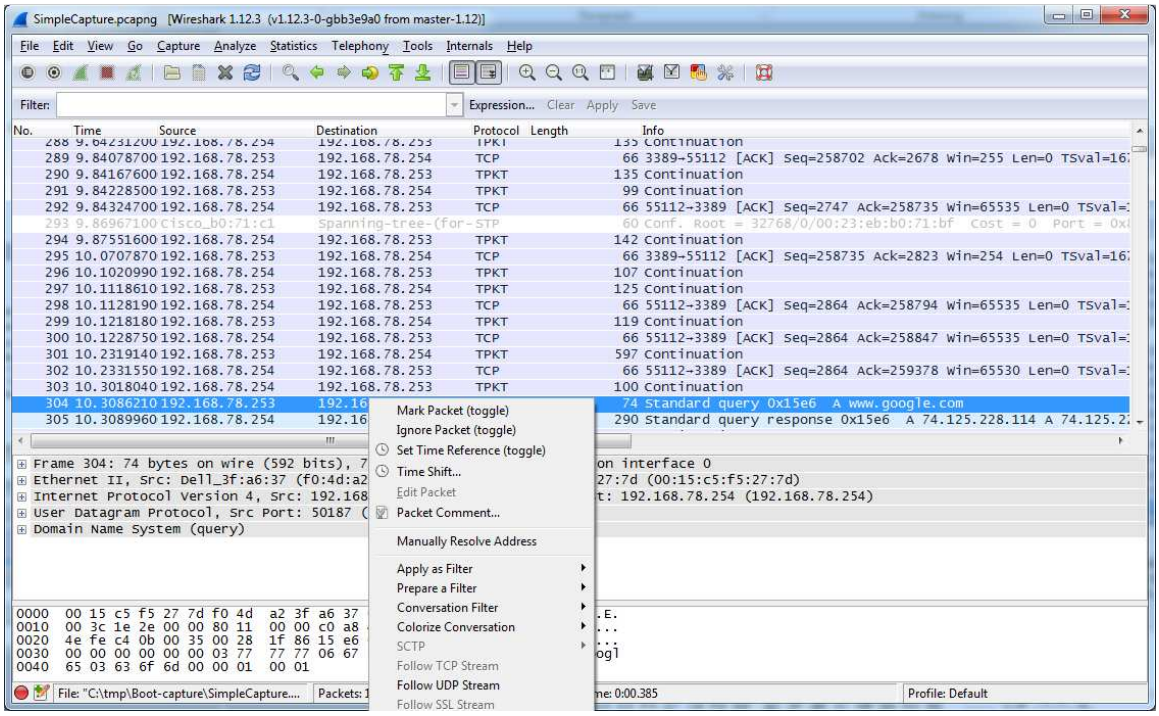
Click on *File* → *Open* and select the **SimpleCapture.pcap** file located on your desktop.

Wireshark General Layout

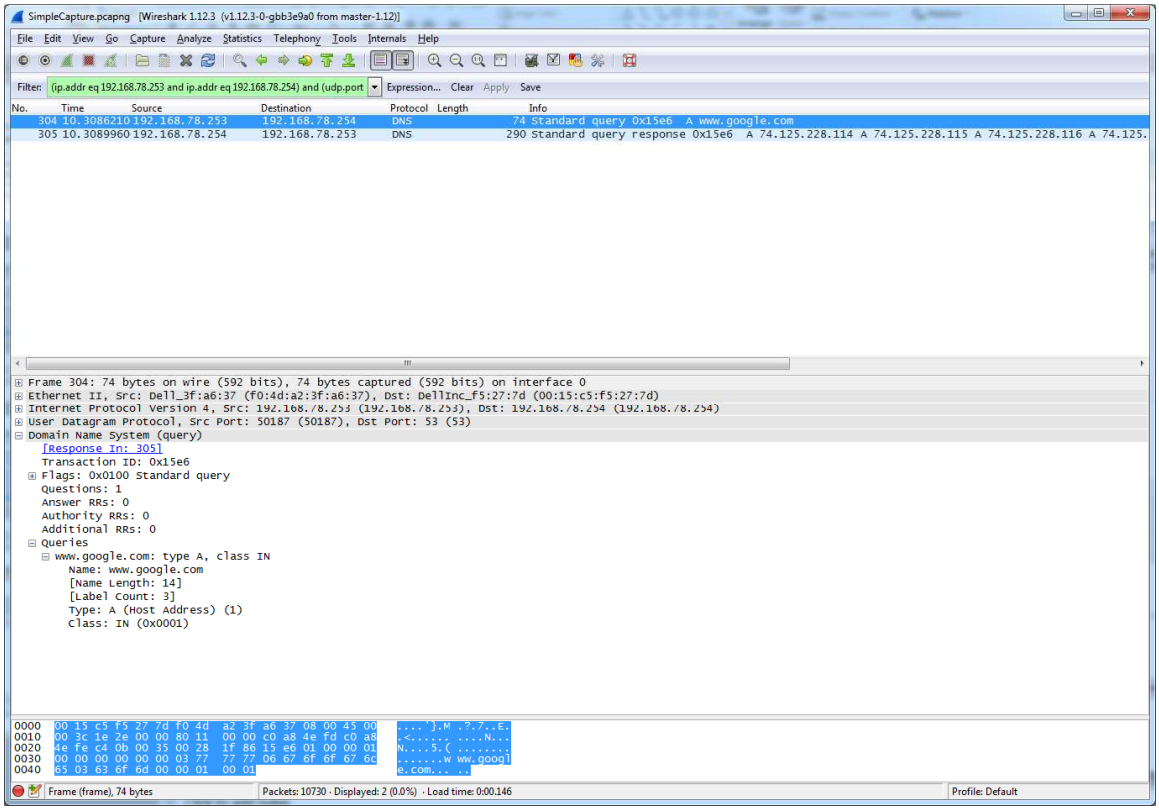
Each line in the capture corresponds to a single packet seen on the network. This is shown in the top pane. The default display shows the time of the packet (relative to the start of the capture) as well as the source and destination IP addresses, the protocol used and some information about the packet. You can click on a row to obtain more information. This allows the other windows to be used. The middle pane contains more internal details on the packet selected in the top frame. These can be expanded out into varying levels of detail. The bottom screen displays the actual data. On the left-hand side you see the hexadecimal representation of the data. On the right-hand side the character representation is displayed. Note the headings displayed in the first section. The first column denotes the packet number. The second column is the time relative to the start of the capture. The remaining columns are the Source IP address, the Destination IP address, the Network Protocol, the Packet Length and Information about the packet.

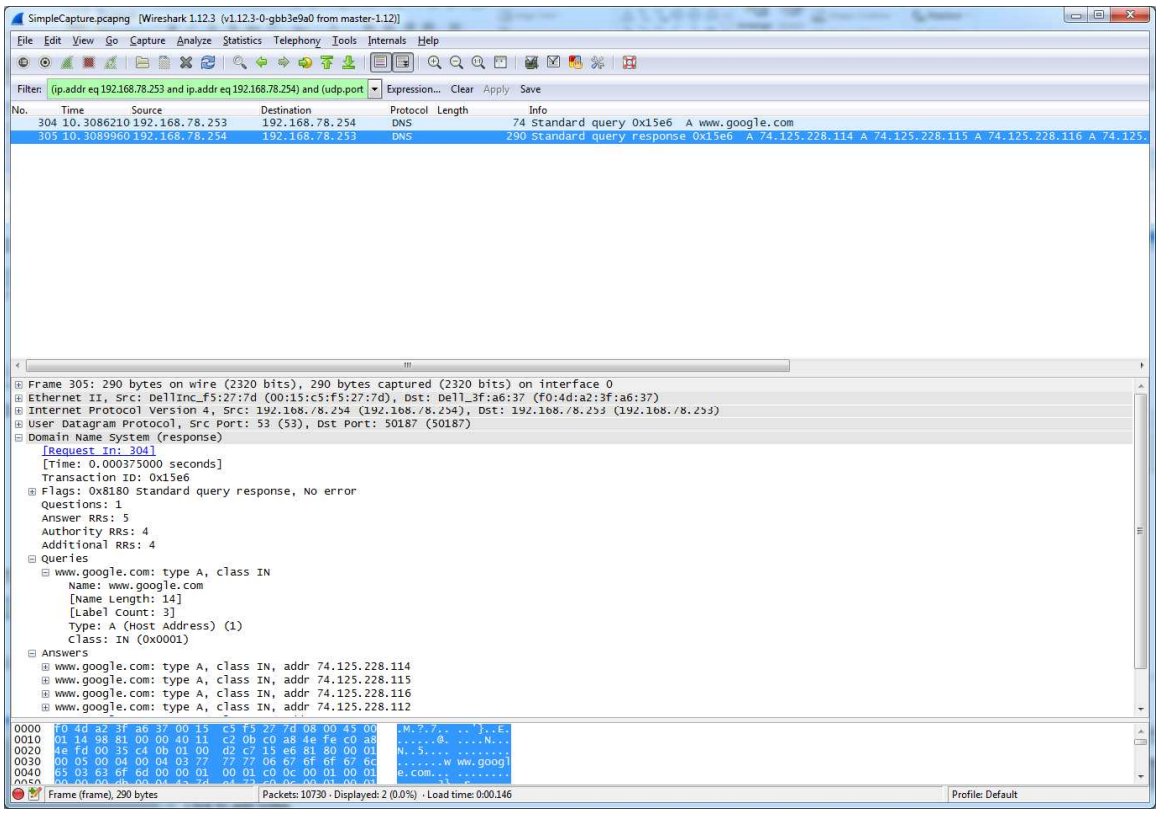


We will begin by examining the sequence of events that take place when a user performs a DNS query. DNS stands for *D*omain *N*ame *S*ervices. DNS takes a common fully qualified domain name and translates it to a corresponding Internet address. The sequences of events that take place are first a request is made from a source machine to a DNS server. If the server recognizes the name requested it sends a response with the IP address associated with the name. Two possible situations can occur if the server does not know the name requested. These situations depend upon how the request is configured. If the request is a recursive request then the source machine will depend on the server to forward on the request to find the answer. If the request is setup to be iterative then the server will respond that it does not know the name and the source machine will need to make a request to another server. Let's take a look at a simple request that was made in our network capture. In our example a user has made a request via the *nslookup* command to get the ip address for ***www.google.com.*** To view this traffic in Wireshark, scroll down and select packet # ***304***. *Right-click* on the packet and select *Follow UDP Stream*

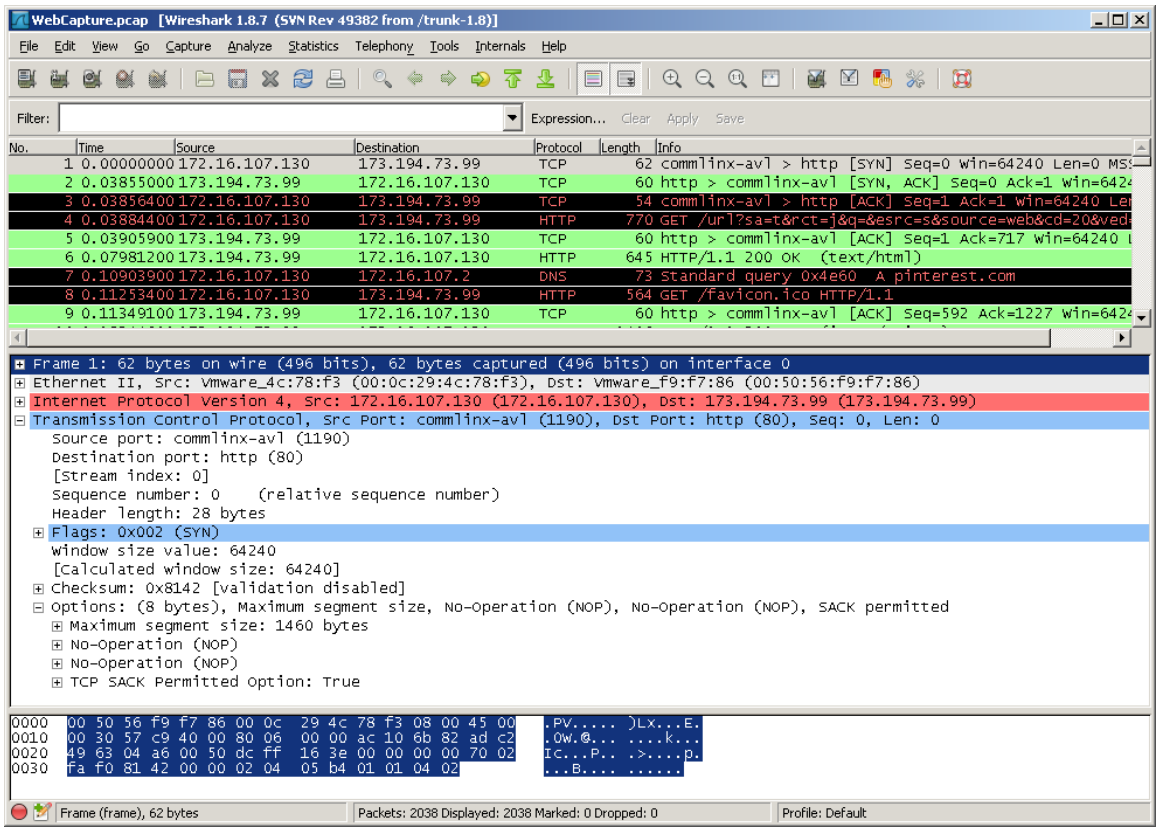


A Follow UDP Stream will appear. You will notice that the Stream Content contains www.google.com and other nonsensical characters. Close this window and return to the main Wireshark window. Notice now that only two packets appear. Click on the first packet and look in the second pane. If you look in the flags section you will notice that this is a recursive query. This tells us that the server will send on the request if it does not have an answer.


SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: (ip.addr eq 192.168.78.253 and ip.addr eq 192.168.78.254) and (udp.port
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
304 10.3086210 192.168.78.253 192.168.78.254 DNS 74 Standard query 0x15e6 A www.google.com
305 10.3089960 192.168.78.254 192.168.78.253 DNS 290 Standard query response 0x15e6 A 74.125.228.114 A 74.125.228.115 A 74.125.228.116 A 74.125.
Frame 304: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: Dell_3f:a6:37 (f0:4d:a2:3f:a6:37), Dst: DellInc_f5:27:7d (00:15:c5:f5:27:7d)
Internet Protocol Version 4, Src: 192.168.78.253 (192.168.78.253), Dst: 192.168.78.254 (192.168.78.254)
User Datagram Protocol, Src Port: 50187 (50187), Dst Port: 53 (53)
Domain Name System (query)
[Response In: 305]
Transaction ID: 0x15e6
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.google.com: type A, class IN
Name: www.google.com
[Name Length: 14]
[Label Count: 3]
Type: A (Host Address) (1)
Class: IN (0x0001)
0000 00 15 c5 f5 27 7d f0 4d a2 3f a6 37 08 00 45 00 ....'}.M .?.7..E.
0010 00 3c 1e 2e 00 00 80 11 00 00 c0 a8 4e fd c0 a8 .<..... ....N...
0020 4e fe c4 0b 00 35 00 28 1f 86 15 e6 01 00 00 01 N....5.(........
0030 00 00 00 00 00 00 03 77 77 77 06 67 6f 6f 67 6c ....w ww.googl
0040 65 03 63 6f 6d 00 00 01 00 01 e.com... ..
Frame (frame), 74 bytes
Packets: 10730 · Displayed: 2 (0.0%) · Load time: 0:00.146
Profile: Default


SimpleCapture.pcapng [Wireshark 1.12.3 (v1.12.3-0-gbb3e9a0 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: (ip.addr eq 192.168.78.253 and ip.addr eq 192.168.78.254) and (udp.port
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
304 10.3086210 192.168.78.253 192.168.78.254 DNS 74 Standard query 0x15e6 A www.google.com
305 10.3089960 192.168.78.254 192.168.78.253 DNS 290 Standard query response 0x15e6 A 74.125.228.114 A 74.125.228.115 A 74.125.228.116 A 74.125.
Frame 305: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface 0
Ethernet II, Src: DellInc_f5:27:7d (00:15:c5:f5:27:7d), Dst: Dell_3f:a6:37 (f0:4d:a2:3f:a6:37)
Internet Protocol Version 4, Src: 192.168.78.254 (192.168.78.254), Dst: 192.168.78.253 (192.168.78.253)
User Datagram Protocol, Src Port: 53 (53), Dst Port: 50187 (50187)
Domain Name System (response)
[Request In: 304]
[Time: 0.000375000 seconds]
Transaction ID: 0x15e6
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 5
Authority RRs: 4
Additional RRs: 4
Queries
www.google.com: type A, class IN
Name: www.google.com
[Name Length: 14]
[Label Count: 3]
Type: A (Host Address) (1)
Class: IN (0x0001)
Answers
www.google.com: type A, class IN, addr 74.125.228.114
www.google.com: type A, class IN, addr 74.125.228.115
www.google.com: type A, class IN, addr 74.125.228.116
www.google.com: type A, class IN, addr 74.125.228.112
0000 f0 4d a2 3f a6 37 00 15 c5 f5 27 7d 08 00 45 00 .M.?.7.. ..'}..E.
0010 01 14 98 81 00 00 40 11 c2 0b c0 a8 4e fe c0 a8 ......@. ....N...
0020 4e fd 00 35 c4 0b 01 00 d2 c7 15 e6 81 80 00 01 N..5.... ....
0030 00 05 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c ....w ww.googl
0040 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 01 e.com... ....
Frame (frame), 290 bytes
Packets: 10730 · Displayed: 2 (0.0%) · Load time: 0:00.146
Profile: Default

A *three-way handshake* is used to initiate communication between two machines. When a source machine wants to communicate with a destination machine it will start by sending a SYN request. This tells the destination machine that a conversation is being requested. If the destination machine accepts the request it will respond with a SYN, ACK. When the initial machine receives this response it in turn responds with an ACK response and the conversation begins. Lets close our current packet capture and open another to observe this traffic. Go to *File* → *Close* to close the current capture. Then choose *File* → *Open* and select the file named **WebCapture.pcap**.



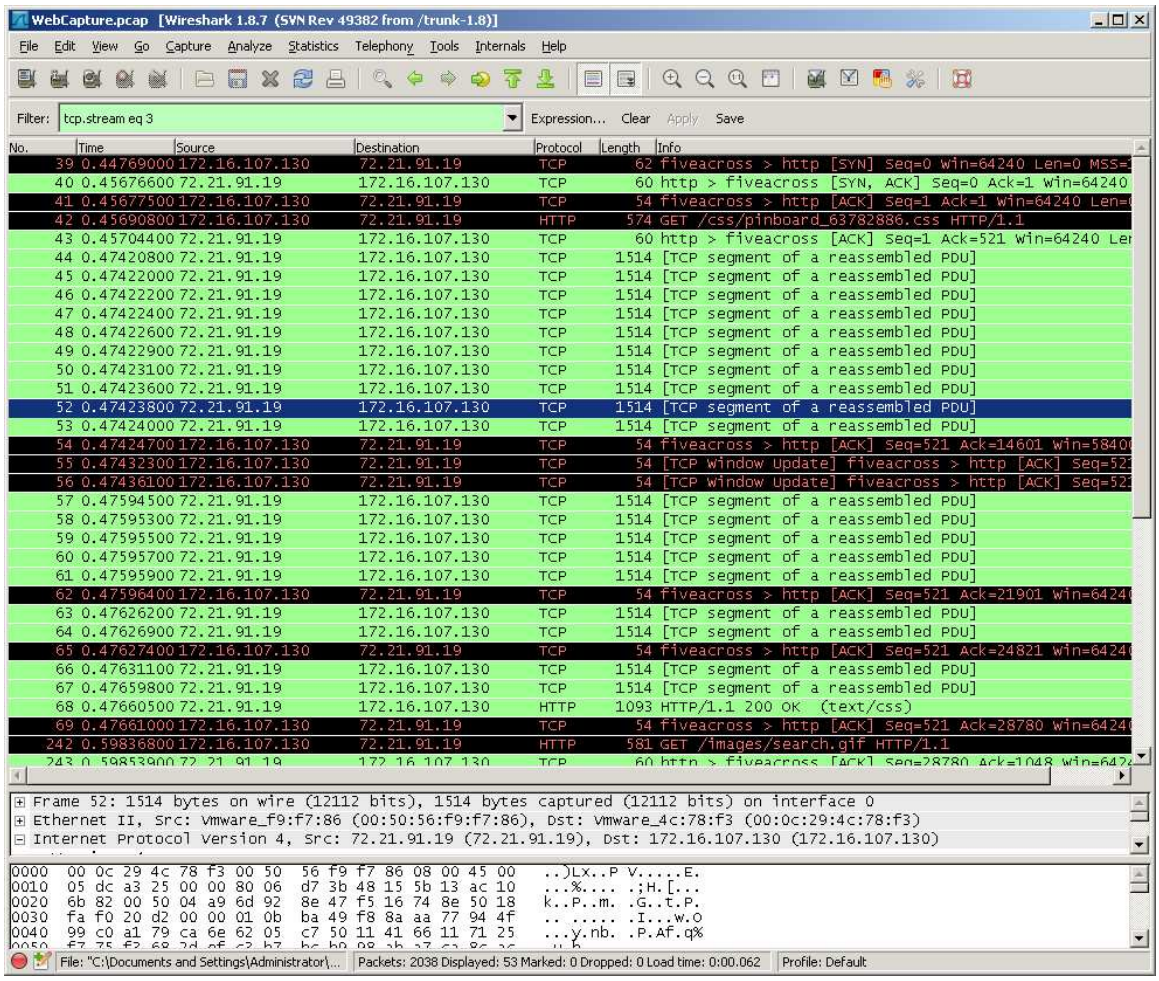
This is a capture of a simple web search. The user has opened up the Google search engine and ran a search for "cute puppy pictures". The user then chose the Pinterest webpage, selected and downloaded a picture.

Lets take a look at the packet capture in detail. In the first packet we see that the source computer (172.16.107.130) sends a SYN request to the destination computer (173.194.73.99). If you expand the Transmission Control Protocol setion of the second pane you can see that the source port is 1190 and the destination port is 80 (indicating an http request is coming). The Flags section shows that this is an initial SYN request. The next packet displays the SYN,ACK response coming back from 173.194.73.99 port 80 to

172.16.107.130 port 1190. Packet 3 shows the final ACK response completing the three-way handshake.

Packet 4 marks the beginning of the "cute puppy pictures" search. Packet 7 shows the DNS lookup (query) for pinterest.com (the site where the puppy picture resides). The DNS response is seen in packet 11. Packets 12-17 indicate two different three-way handshakes for requests from pinterest.

Scroll down and click on packet 42. Right click and choose Follow TCP Stream. Notice the two sets of headers followed by a bunch of confusing information. This confusing information is the binary picture being downloaded. If you close the Follow TCP Stream window you will notice a filter has been entered in the filter section (we will talk more about filters later). What is important to know at this point is that this request has filtered out all other packets and now we can follow just this network conversation.

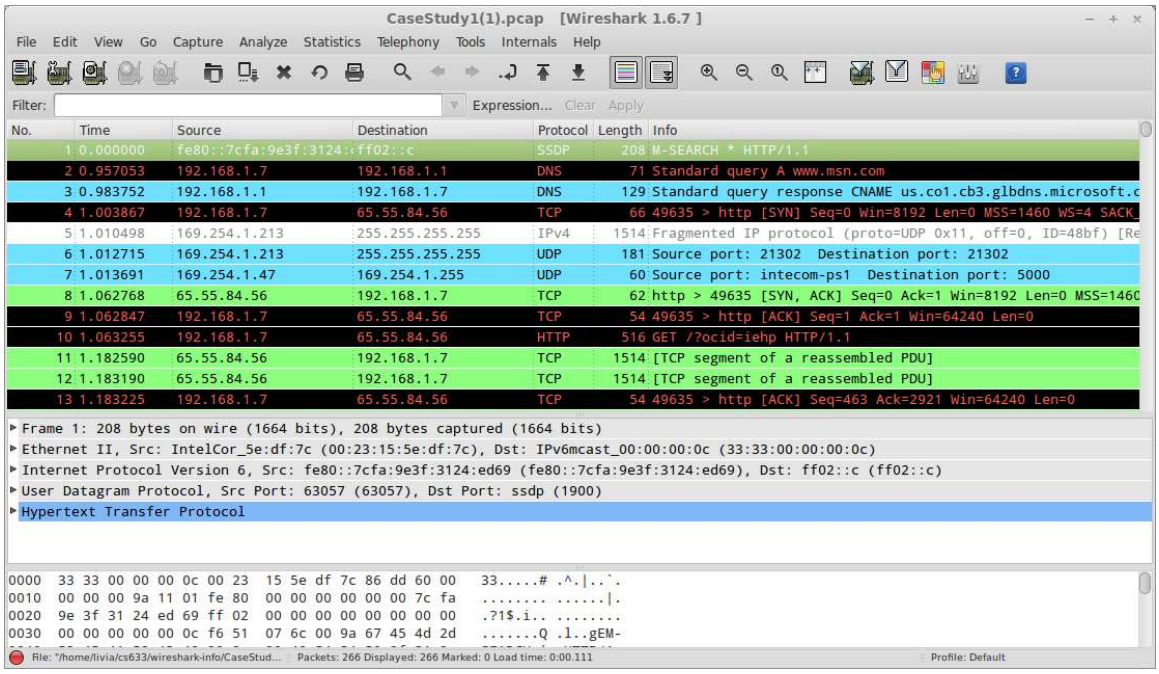


This section basically shows the downloading of the image. Notice all the lines which contain "15114 [TCP segment of a reassembled PDU]". This is portions of the image being downloaded.

This concludes this exercise close the Capture file and procede to Exercise 2.

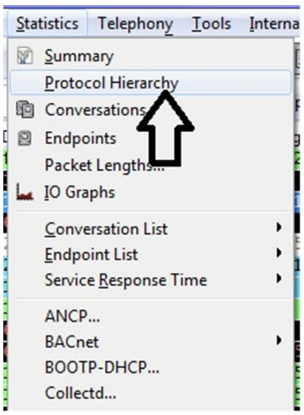
Exercise 2 - Gathering General Information and Statistics and More Wireshark Analysis

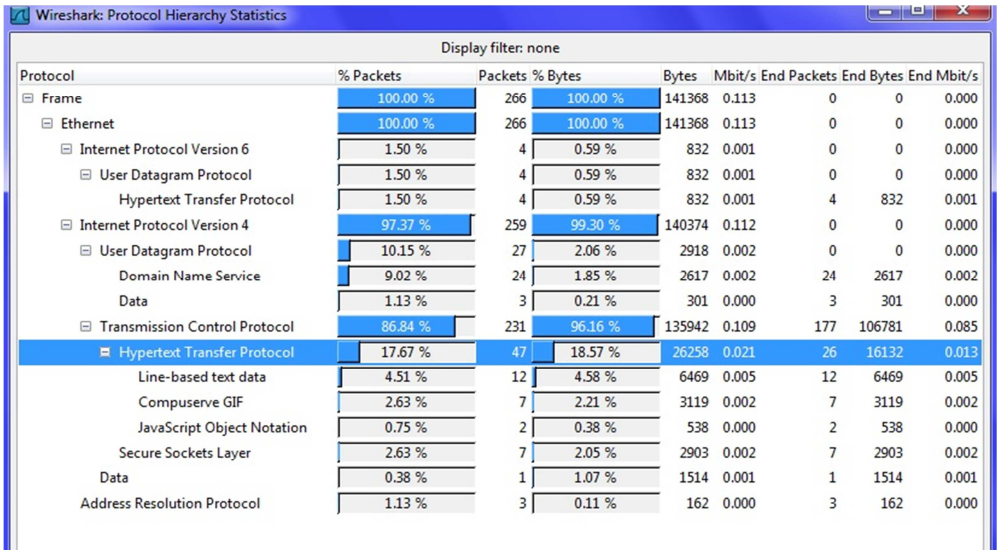
Click on *File* → *Open* and select the **CaseStudy1.pcap** file located on your desktop. You should now have a window displaying that is similar to below:



To determine what type of data has been captured in this file we can to go the *Statistics* section and select

Protocol Hierarchy

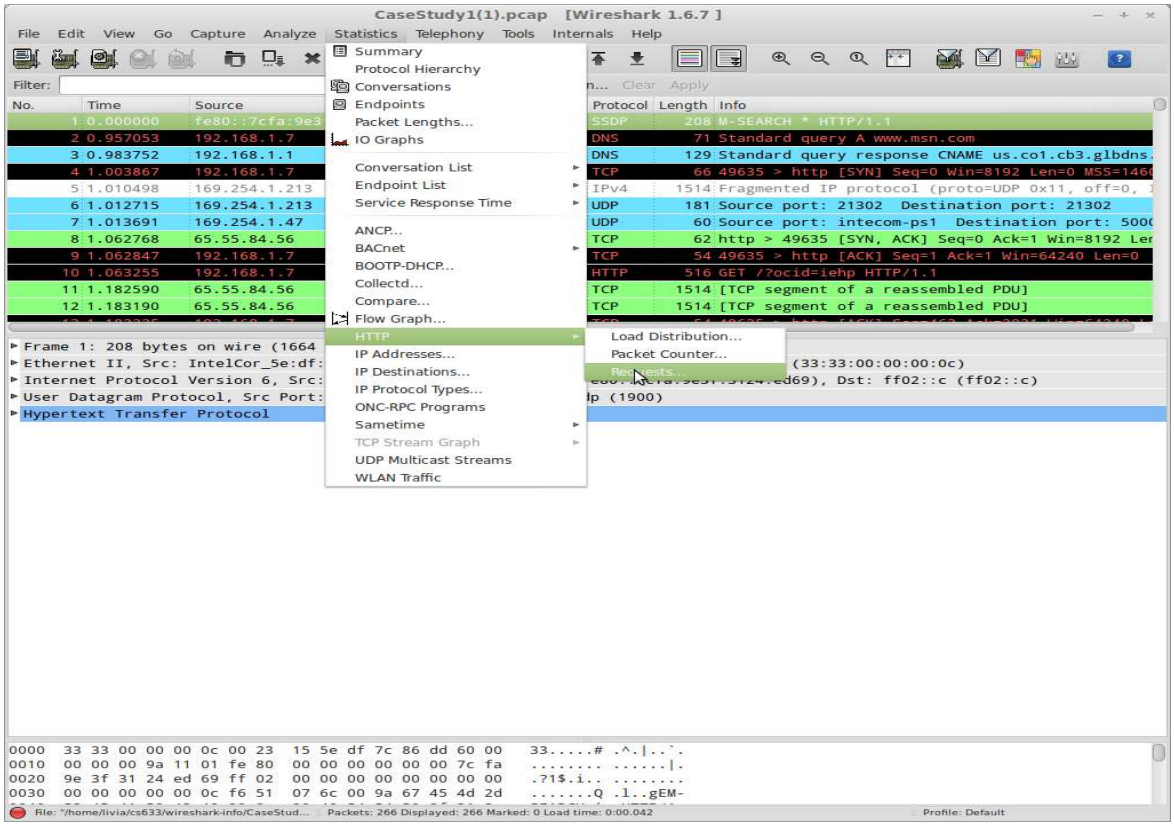


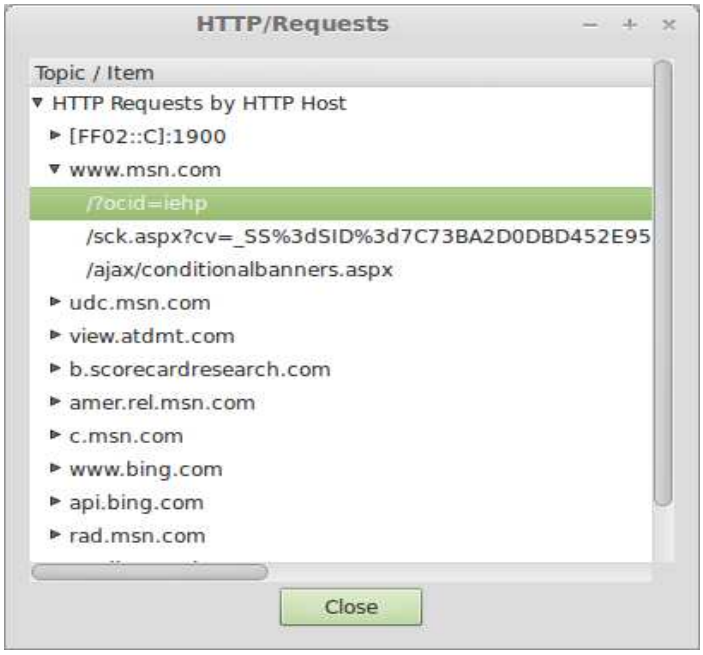


Notice the line highlighted above.
This shows that 17.67% of the traffic contained in this capture is Hypertext Transfer protocol (http – web traffic). This may not seem like a lot but you must remember that one web request will generate several packets of data.

Other interesting statistics can be gathered easily through selections made under the *Statistics* section. We will examine a few briefly.

Close your current window and select the Statistics section again. This time choose *HTTP → Requests…* Leave the filter blank and click on Create Stat





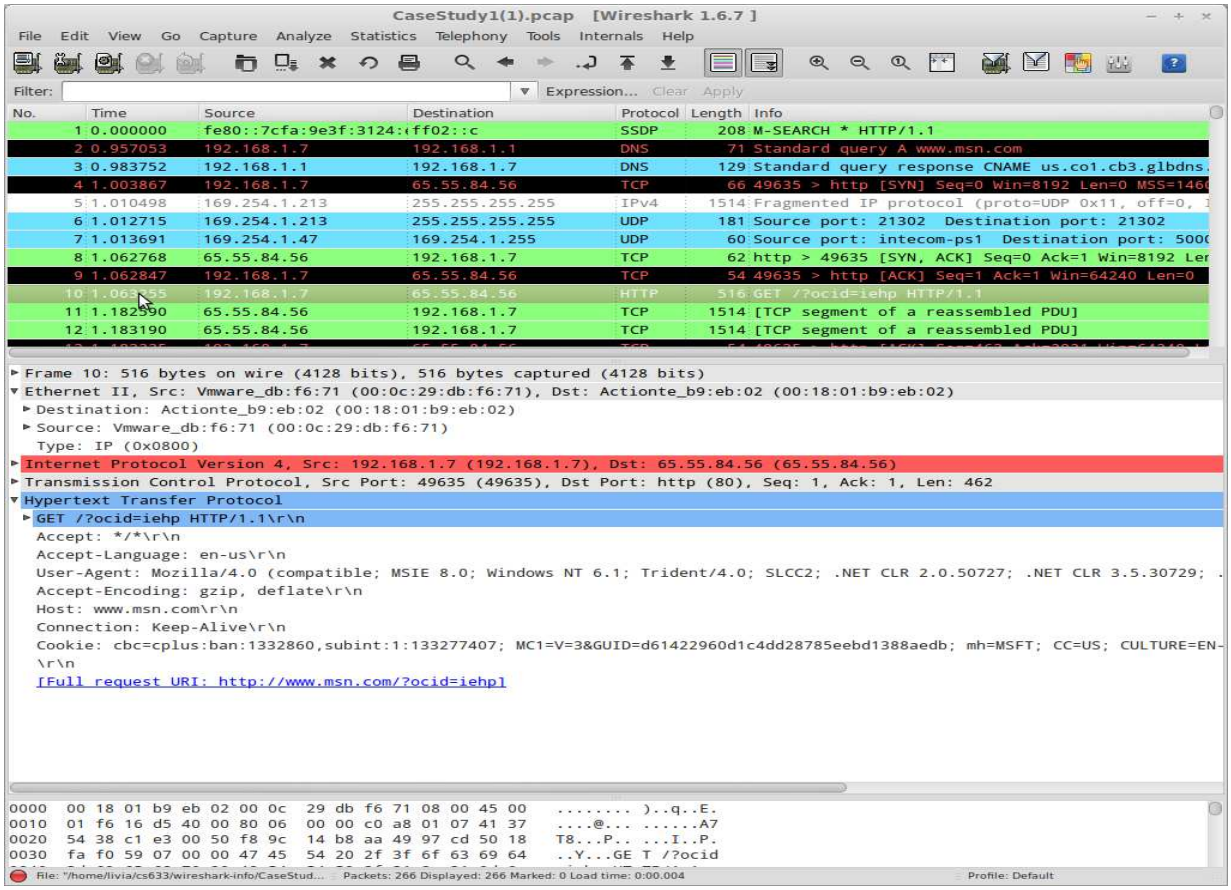
(The above figure might be obsolete and what you see might be a little bit different)

This list now contains all the http requests made.

Wireshark includes a complex color-coding scheme. The default settings are as follows:

Name	String
Bad TCP	tcp.analysis.flags
HSRP State Change	hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Change	stp.type == 0x80
OSPF State Change	ospf.msg != 1
ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type
ARP	arp
ICMP	icmp || icmpv6
TCP RST	tcp.flags.reset eq 1
SCTP ABORT	sctp.chunk_type eq ABORT
TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim) || (ip.dst == 224.0.0.0/24 && ip.ttl != 1)
Checksum Errors	cdp.checksum_bad==1 || edp.checksum_bad==1 || ip.checksum_bad==1 || tcp.checksum_bad==1 ||
SMB	smb || nbss || nbns || nbipx || ipxsap || netbios
HTTP	http || tcp.port == 80
IPX	ipx || spx
DCERPC	dcerpc
Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || gvrp || igmp || ismp
TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
TCP	tcp
UDP	udp
Broadcast	eth[0] & 1

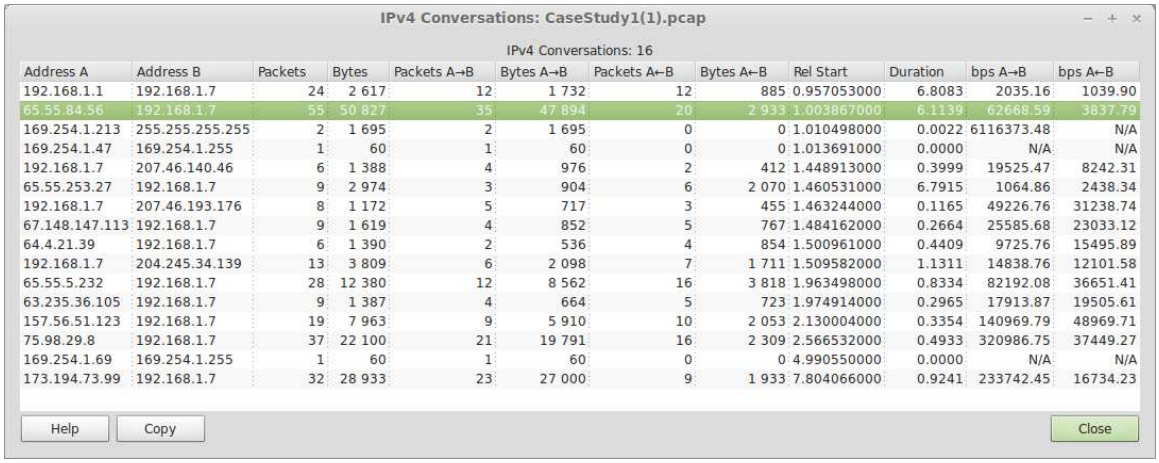
Close this window and lets analyze the first http packet – click on line 10.



Examine the information displayed in the middle pane of the windows. Wireshark does a great job of showing all the details of the packet (including the lower levels and not just the application layer). In this case, we get the most information from the application layer. Here we can tell that the HTTP packet is a GET request for the /?ocid=iehp page of www.msn.com. Wireshark even summarizes the full request below to be http://www.msn.com/?ocid=iehp.

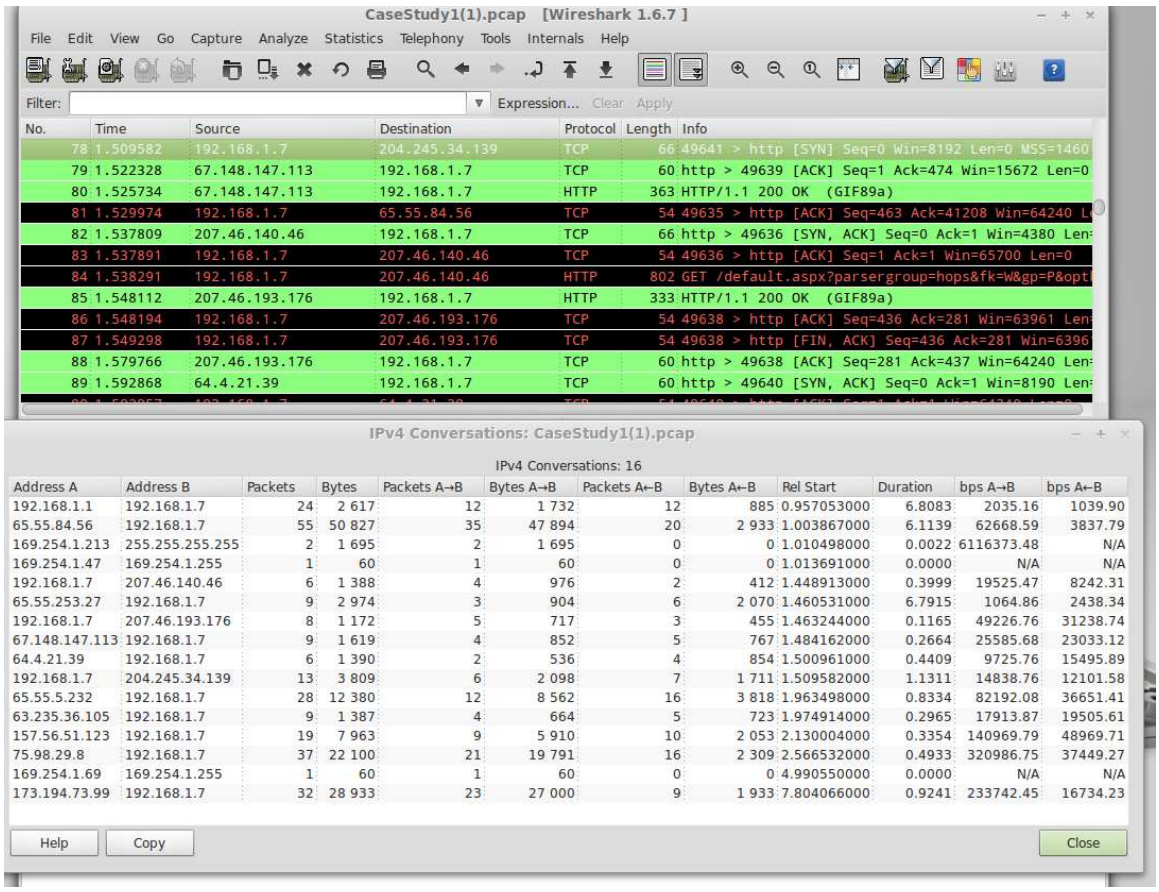
Conversations

A network conversation is the traffic between two specific endpoints. Along with the addresses, packet counters, and byte counters, this window also has the time in seconds between the start of the capture and the start of the conversation ("Rel Start"), the duration of the conversation in seconds, and the average bits (not bytes) per second in each direction. Lets take a look. Click on the Statistics section and go to Conversation



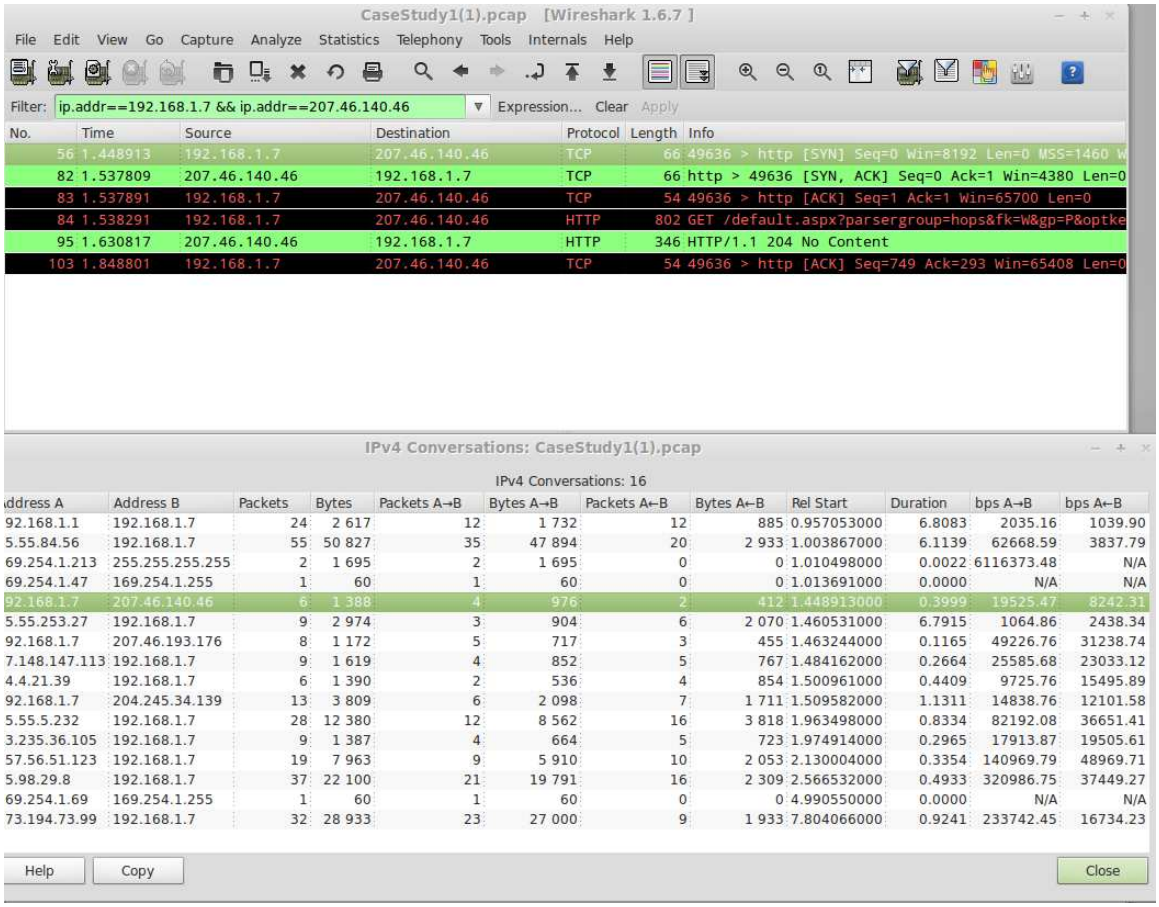
list → IPv4. A window similar to that below should appear.

Each row in the list shows the statistical values for exactly one conversation. Conversations can be further shown at each of the different levels. From the conversation window, filters can also be applied. To demonstrate this situate this window so that it is so that you can see both it and the top pane of the main Wireshark window as is demonstrated below:

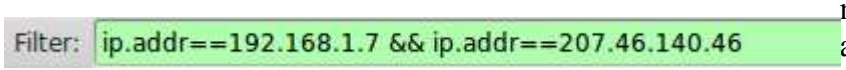


Since we have created no filters all the network traffic appears. Choose the fifth line down in the conversation window

Right-click on the line and choose Apply Filter then Selected then A ↔ B. Notice how the contents of the first pane of the main Wireshark Window has changed. Now you are viewing only the traffic which transpired between the source IP address of 192.168.1.7 and the destination address of 207.46.140.46



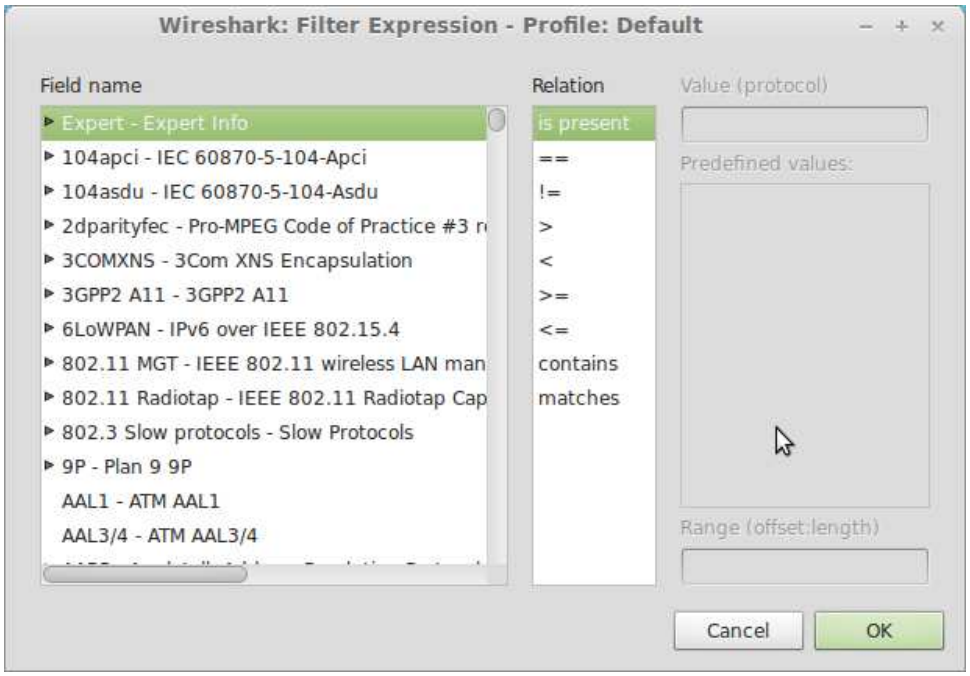
Notice that the Filter section above the first pane has been filled in. Close the Conversation window and lets examine this further. The Filter that we applied was:



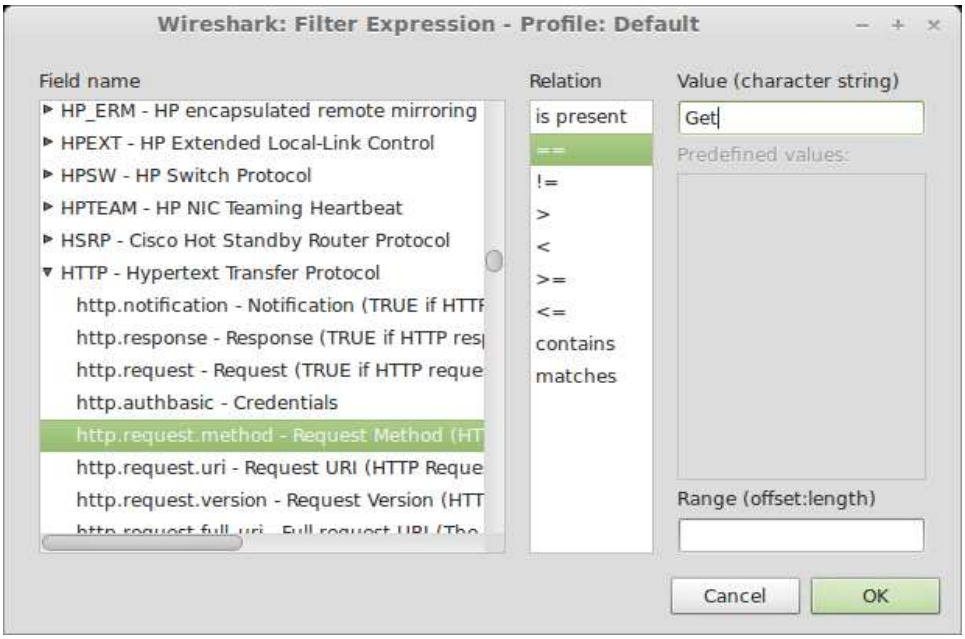
This was one easy way to filter out all traffic except that between IP address 192.168.1.7 and IP address 207.46.140.46.

Filters are a good way to decipher through all the packets and zone in on specific information. Let's examine a few simple filters that we can make through the use of the *Expression...* builder. Press the *Clear* button located to the Filter and then press the *Expression...* key.

It may take a few seconds but a Filter Expression window should appear



Here you can build your own filters. Scroll down to ***Http*** in the *Field Name* section and expand the options. Notice that there are many different sub-filters that you can use to fine tune your search. Lets narrow our search to all the Get requests. Click on the sub-filter labeled *http.request.method.* In the Relation section choose the = = "equal" sign. Notice that the Value section becomes active. Type in **GET** (be sure to use all CAPITAL Letters) and click *OK*



Notice that the filter has been filled in:

Click on Apply and notice the packets that have been selected. This is a list of all the Get requests that have been made in the captured session.

This ends the first exercise. Click *Clear* on the Filter line to exist the filter. Then go to the *File* tab and select *Close* to close the ***CaseStudy1.pcap*** file.

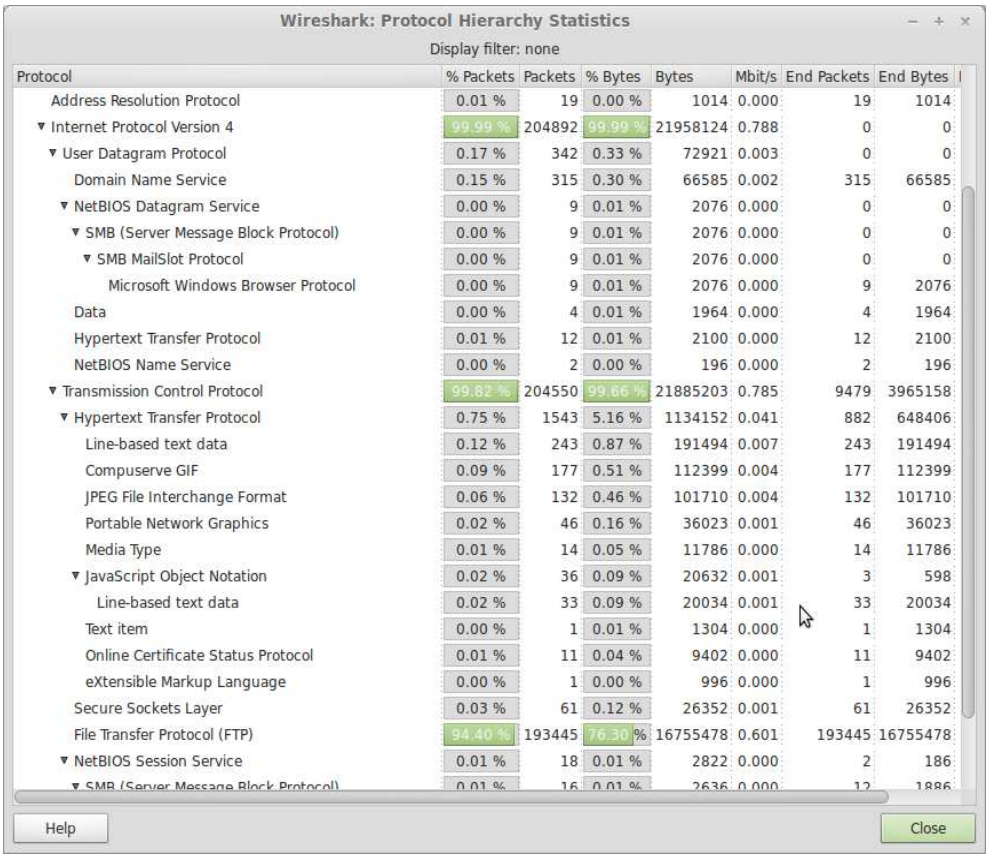
Exercise 2 – More Traffic searches and Filters

Go to the *File* → *Open* tab and open the file located on your desktop named ***CaseStudy2.pcap***

In this packet capture you will immediately notice that there is a lot more traffic. Lets start by looking at the *Protocol Hierarchy* statistics to see if we can gather some

information about what events have taken place. If you remember from the previous exercise this can be done by selecting *Statistics → Protocol Hierarchy*

If you enlarge the window you will notice that **94.4%** of the Packet traffic took involves the **File Transfer Protocol (FTP)**

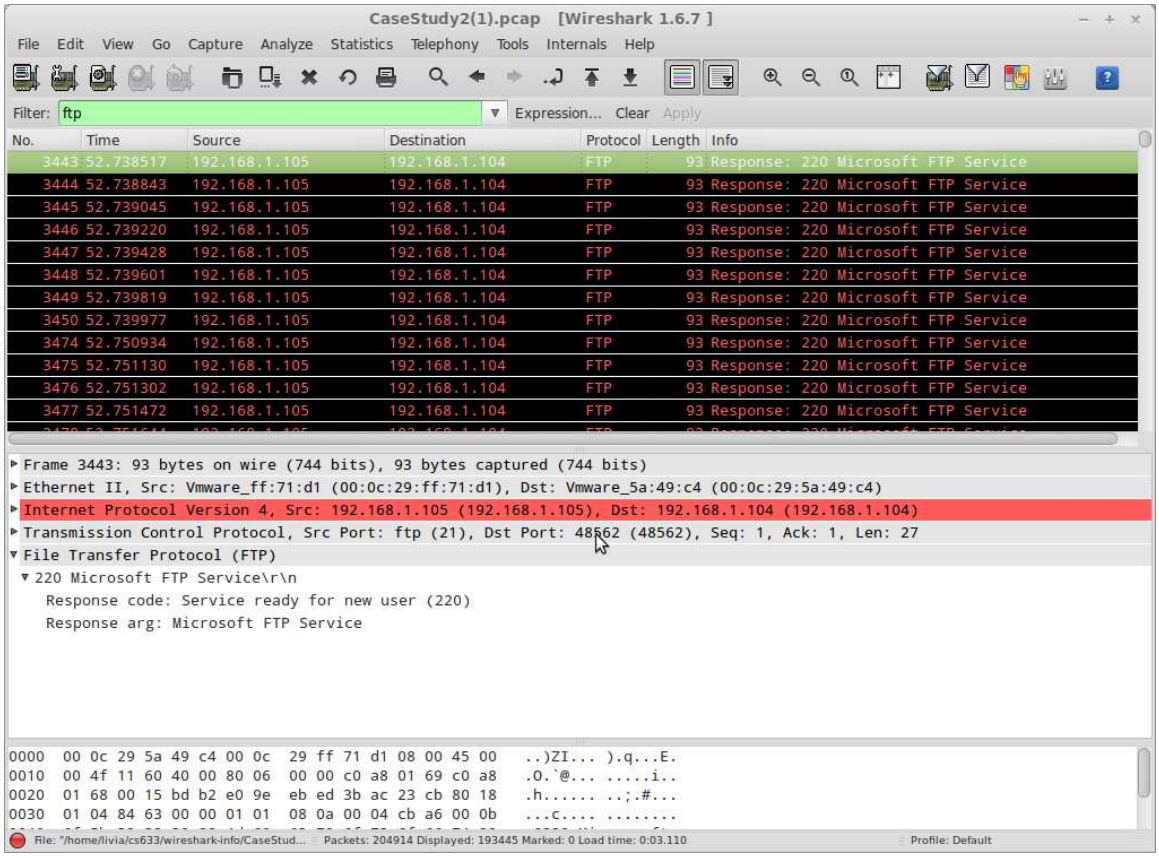


Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes
Address Resolution Protocol	0.01 %	19	0.00 %	1014	0.000	19	1014
▾ Internet Protocol Version 4	99.99 %	204892	99.99 %	21958124	0.788	0	0
▾ User Datagram Protocol	0.17 %	342	0.33 %	72921	0.003	0	0
Domain Name Service	0.15 %	315	0.30 %	66585	0.002	315	66585
▾ NetBIOS Datagram Service	0.00 %	9	0.01 %	2076	0.000	0	0
▾ SMB (Server Message Block Protocol)	0.00 %	9	0.01 %	2076	0.000	0	0
▾ SMB MailSlot Protocol	0.00 %	9	0.01 %	2076	0.000	0	0
Microsoft Windows Browser Protocol	0.00 %	9	0.01 %	2076	0.000	9	2076
Data	0.00 %	4	0.01 %	1964	0.000	4	1964
Hypertext Transfer Protocol	0.01 %	12	0.01 %	2100	0.000	12	2100
NetBIOS Name Service	0.00 %	2	0.00 %	196	0.000	2	196
▾ Transmission Control Protocol	99.82 %	204550	99.66 %	21885203	0.785	9479	3965158
▾ Hypertext Transfer Protocol	0.75 %	1543	5.16 %	1134152	0.041	882	648406
Line-based text data	0.12 %	243	0.87 %	191494	0.007	243	191494
Compuserve GIF	0.09 %	177	0.51 %	112399	0.004	177	112399
JPEG File Interchange Format	0.06 %	132	0.46 %	101710	0.004	132	101710
Portable Network Graphics	0.02 %	46	0.16 %	36023	0.001	46	36023
Media Type	0.01 %	14	0.05 %	11786	0.000	14	11786
▾ JavaScript Object Notation	0.02 %	36	0.09 %	20632	0.001	3	598
Line-based text data	0.02 %	33	0.09 %	20034	0.001	33	20034
Text item	0.00 %	1	0.01 %	1304	0.000	1	1304
Online Certificate Status Protocol	0.01 %	11	0.04 %	9402	0.000	11	9402
eXtensible Markup Language	0.00 %	1	0.00 %	996	0.000	1	996
Secure Sockets Layer	0.03 %	61	0.12 %	26352	0.001	61	26352
File Transfer Protocol (FTP)	94.40 %	193445	76.30 %	16755478	0.601	193445	16755478
▾ NetBIOS Session Service	0.01 %	18	0.01 %	2822	0.000	2	186
▾ SMB (Server Message Block Protocol)	0.01 %	16	0.01 %	2636	0.000	12	1886

This amount of FTP traffic in one network traffic seems very suspicious and warrants closer examination. To do this we need to apply a filter. This can be simply done by typing the word ftp in the Filter Section of the main Window and clicking apply

This traffic appears even more suspicious since at first glance it appears that there are multiple repeated packets. This does not resemble the expected traffic of a legitimate FTP request. Lets examine some of the packets in more detail. Click on one of the first

packets in the Window and fully expand the File Transfer Protocol (FTP) section in the second pane

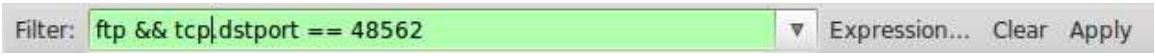


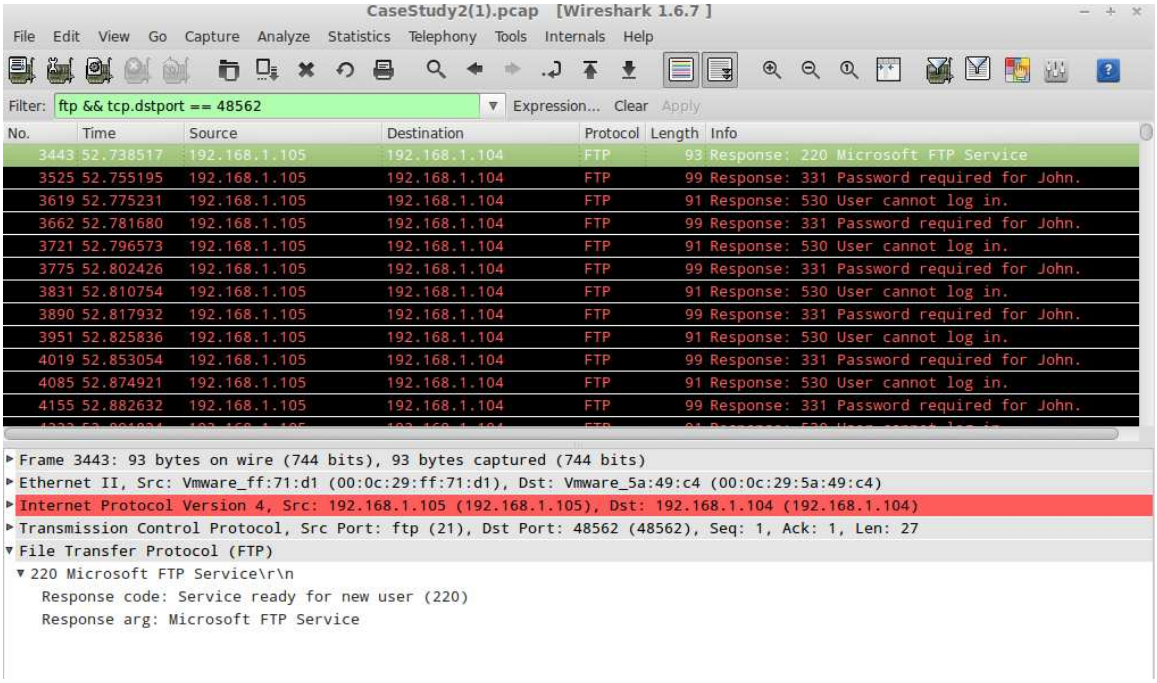
From this we gather that it appears that FTP is ready for a new user. This seems to suggest multiple sessions attempting to be opened and that they are all being attempted on same IP address. Click on the next several packets and take note of the Destination Port. This is pointed to by the the pointer in the above display and appears in the following line

```
▸ Transmission Control Protocol, Src Port: ftp (21), Dst Port: 48562 (48562),
```

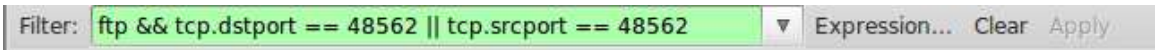
Click on the next Packet in the first frame and notice that the Destination port has increased from 48562 to 48563. Click on the next several packets and again take note of the destination port's incremental changes. This raises another red flag and warrants further investigation. Let's fine tune our filter and see what we can find out. We will filter out all the traffic attempted at one of the destination ports listed. To display a

specific destination port we need to use the tcp.dstport filter. Type in the following in the Filter section and select Apply

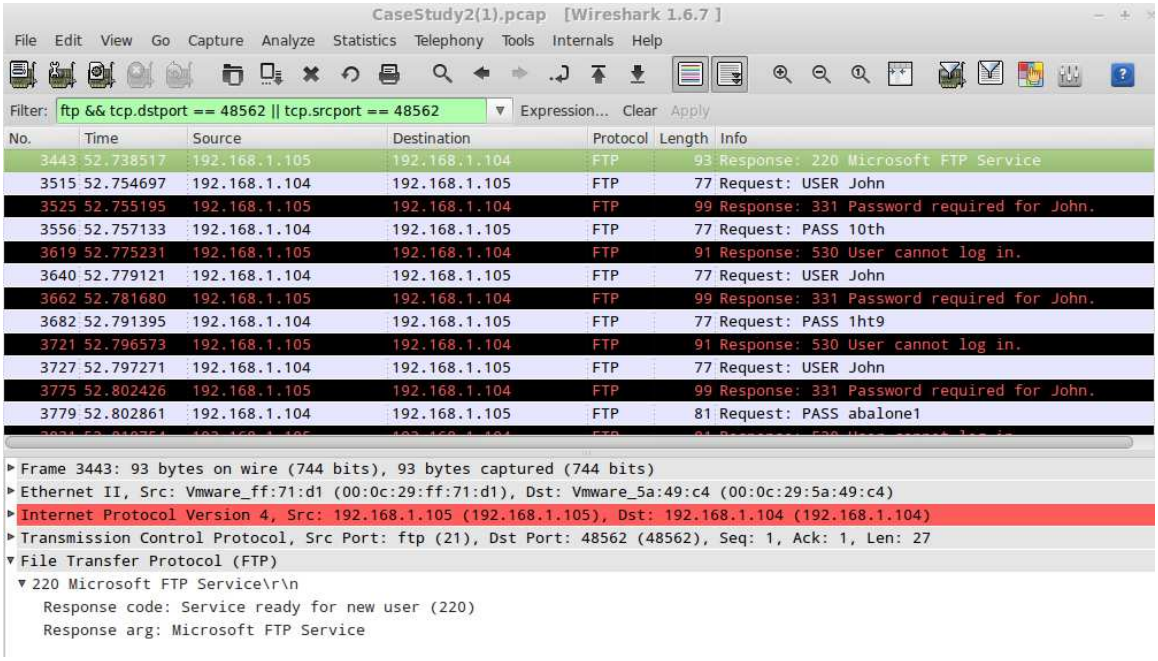




With this filter it seems that someone is trying to login a multitude of time. We are, however, only getting one side of the conversation. This is because with the tcp.dstport setting we are only seeing the return response and not the original request. We can modify our filter to include both sides of the conversation by adding a filter for the source port as follows:

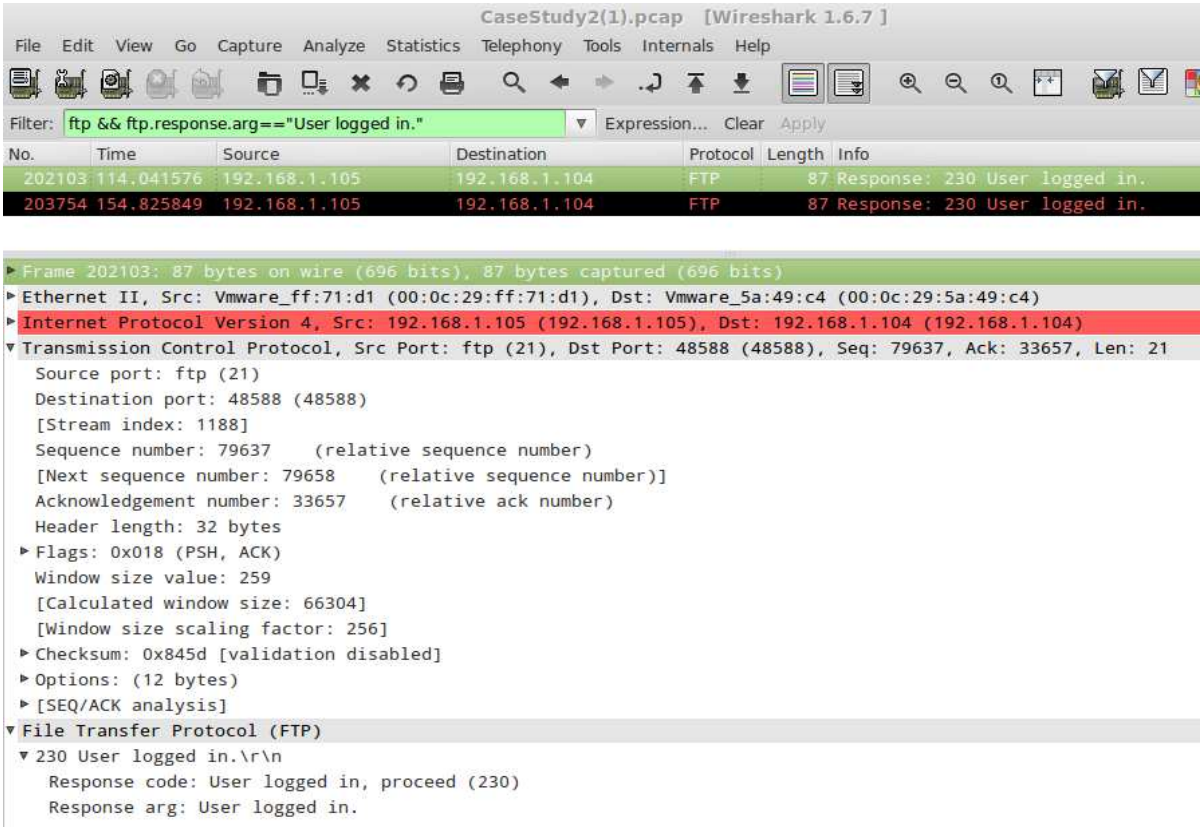


Note the added output:

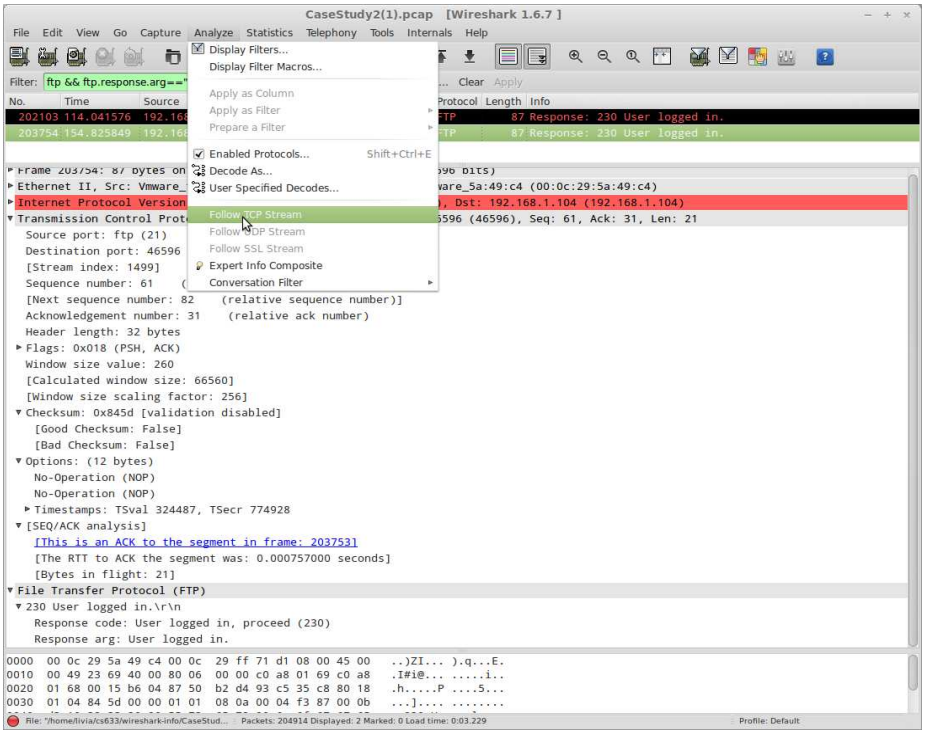


With this information we can finally see what is happening. It appears that this is a password attack against and ftp server. The only thing we do not know at this point is whether the password cracking attack was successful. We can determine this by adding one more filter. When a user successfully logs in the response "User Logged In" is sent. Now we will filter for this response by typing in the filter displayed below (Be sure to type the filter **exactly as it appears** below – including the **period** after the word **in**)

This filter returns two packets



Examining the detail found in the second pane shows that a successful login was made from IP address 192.168.1.105 to 192.168.1.104 via FTP on Dst Port 48588 and again on Dst Port 46596. To determine what login and password were used we can follow the TCP stream. This can be done by selecting one of the packets. Then go up to the *Analyze* label and click on *Follow TCP Stream*



A Follow TCP Stream window will appear. Scroll down to the last entries and you will see that after multiple unsuccessful attempts the User ***John*** logged in successfully using the password ***Password1234***. John had a very weak password and his account was compromised via the use of a dictionary attack. Beyond this, we could look into the commands issued once the connection was established to determine what the attacker did once he obtained access to FTP.

One question might also be brought up. How did the attacker know FTP was enabled? This might suggest a port scan was performed. To check this, we need to perform a filter on some TCP flags. TCP packets have eight flags. They are FIN, SYN, RST, PSH, ACK, URG, ECE and CWR. These flags have decimal numbers assigned to them as follows:

FIN = 1

SYN = 2

RST = 4

PSH = 8

ACK = 16

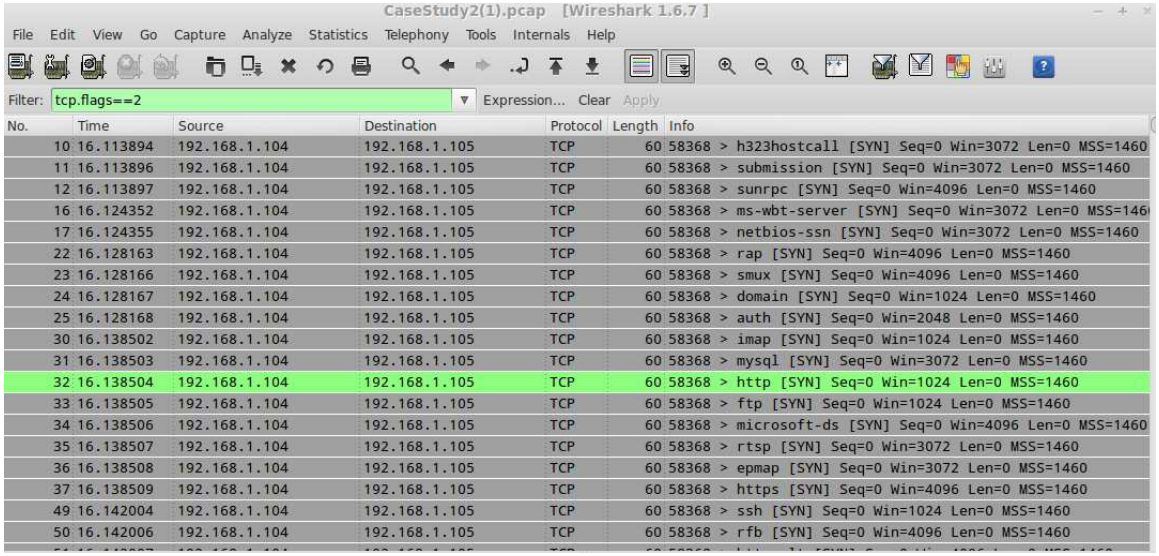
URG = 32

ECE = 64

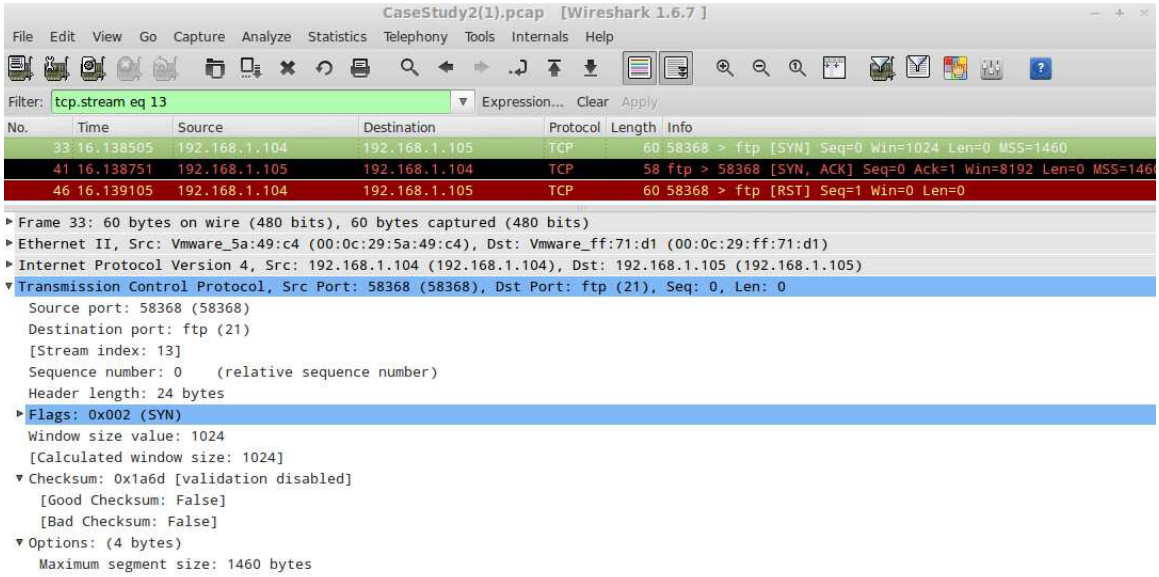
CWR = 129

To check if a SYN/ACK flag is set we add 2 (the SYN value) to 16 (the ACK value) and the result would be 18. A common port scan is a SYN scan, We will first check for that using the following filter:

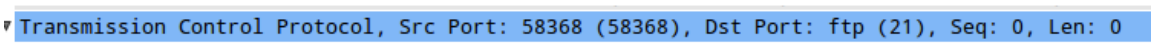
The resulting output indicates that a SYN scan was indeed executed:



This tells us that the attacker knows the ports that were open and that an FTP server was running. Let's examine a few of these lines to see what ports are open. Since we are already aware that the FTP port is open lets scroll down and select line No 33 (see first column in the first pane). Now go to the *Analyze* tab as select *Follow TCP Stream.* Although nothing appears in the TCP Stream window if we close it we see three packets are displayed.



The first line is Packet Number 33 and is the original SYN resquest. The second line (Packet 41) is the SYN, ACK response. This tells the requester that the port is in fact open. Looking in the second pane we see the Transmission Control Protocol line which confirms that the Dst Port is port 21 (FTP).



Clear the filter and lets try another protocol. Start again by typing tcp.flags == 2 in the filter to filter out all SYN requests. This time select packet number 32 for http, and go to *Analyze → Follow TCP Stream.* Here we see that only two packets appear. The original SYN request and then a RST, ACK response. No SYN, ACK is displayed so can determine that this port is not open and no http server is running (or at least not on port 80).

This concludes the Network Analysis exercise. Please close out of the Wireshark application and shutdown your Network Analysis Virtual Machine.

References:

Wireshark Case Study[1,2].pdf presented by Florian Buchholz and Brett Tjaden

Wireshark User Guide

http://www.wireshark.org/docs/wsug_html_chunked/index.html

Advanced Wireshark tutorial: Packet and network security analysis

http://searchsecurity.techtarget.in/tip/Advanced-Wireshark-tutorial-Packet-and-netowrk-security-analysis

Quick and Dirty Wireshark Tutorial

http://searchsecurity.techtarget.in/tutorial/Quick-and-dirty-Wireshark-tutorial