SharkFest'17 US

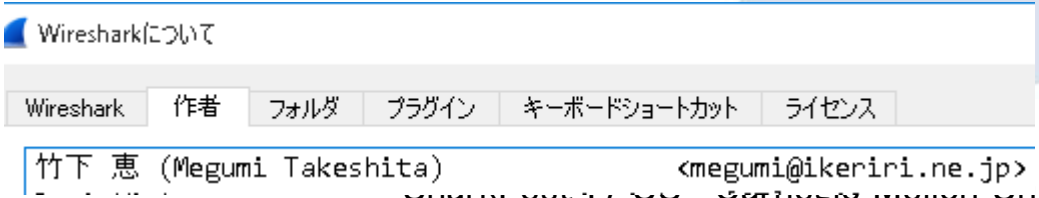
Top 10 Wireshark Tips & Tricks

Sample trace and supplemental files are located
http://www.ikeriri.ne.jp/download/sharkfest

Megumi Takeshita
Packet Otaku | ikeriri network service co.,ltd

Megumi Takeshita, ikeriri network service a.k.a. packet otaku since first Sharkfest

- Old person, bay network,notel network
- Founder, ikeriri network service co.,ltd
- Wrote 10+ books of Wireshark and capturing and network analysis.
  Reseller of CACE technologies, now Riverbed, Metageek, Dualcomm and some in Japan
  Attending all Sharkfest so Congratulations 10th!
- Translator of QT Wireshark into Japanese



23 Top 10 Wireshark TIPS & TRICKS

1. Collect only troubled IEEE802.11 packet
2. Use of Exclude Display filter
3. Stream ID and follow TCP in 2 way
4. Visualize using I/O Graph
5. Capture packet periodically
   and merge for stats
6. Sort and stack the pcap file automatically
   and divide by each filter value
7. Decrypting TLS/SSL data without key pair
8. HTTP2 dissection
9. QUIC dissection
10. Visualize using Elasticsearch ＋ Kibana

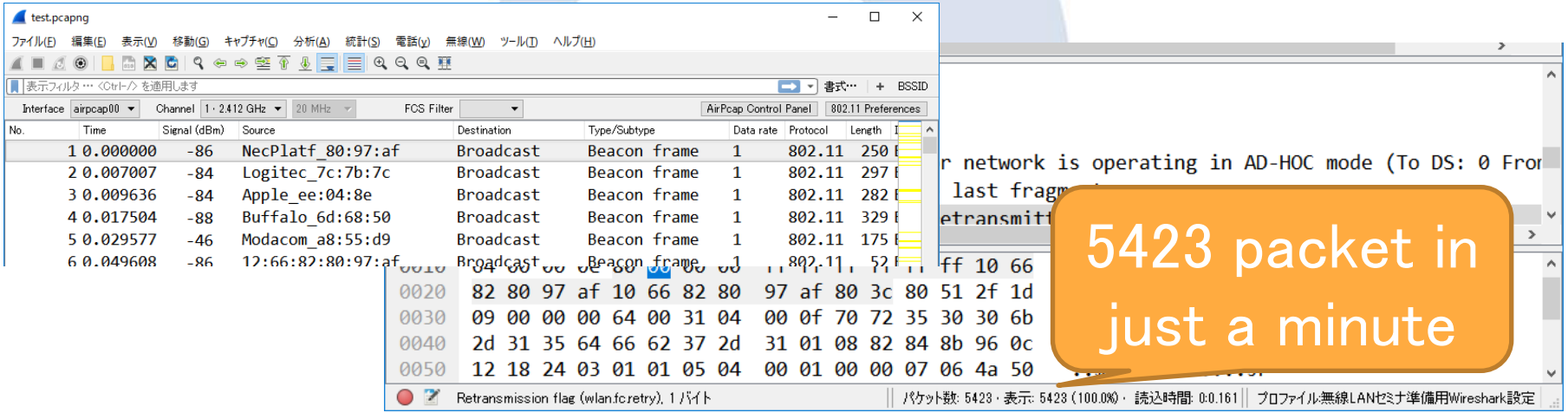
#1. Collect only troubled IEEE802.11 packet

Wireless pcapng file contains tons of other packets

On capturing specified channnel,

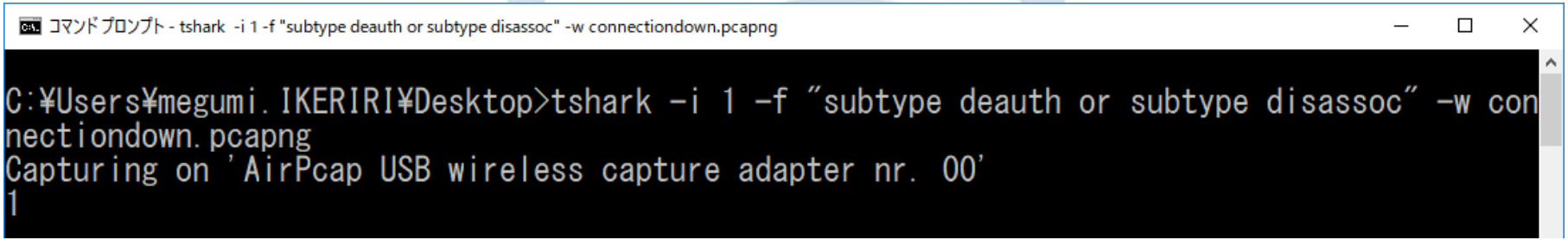
you can collect **only troubled IEEE802.11 packet**

such as Deauthentication and Disassociation.



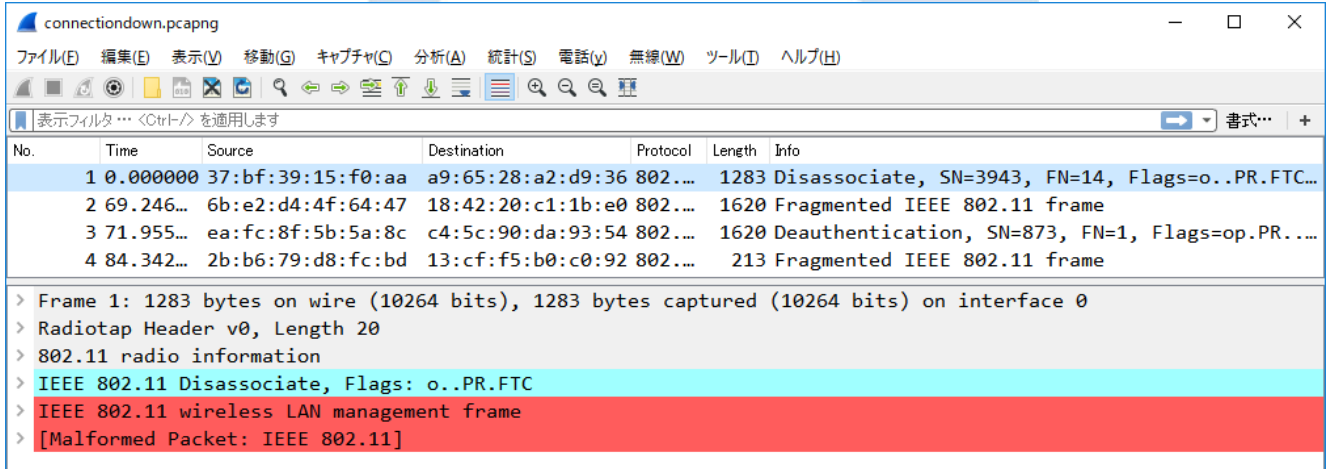
#1. Collect only troubled IEEE802.11 packet

When you collect Deauthentication and Disassociation,
wlan.fc.type_subtype == 12 or wlan.fc.type_subtype==14
In this case capture filter is good for wireless troubleshooting (especially in Tokyo‘s crowded air)
tshark -i 1 -f “**subtype deauth or subtype disassoc**”
(http://www.tcpdump.org/manpages/pcap-filter.7.txt)



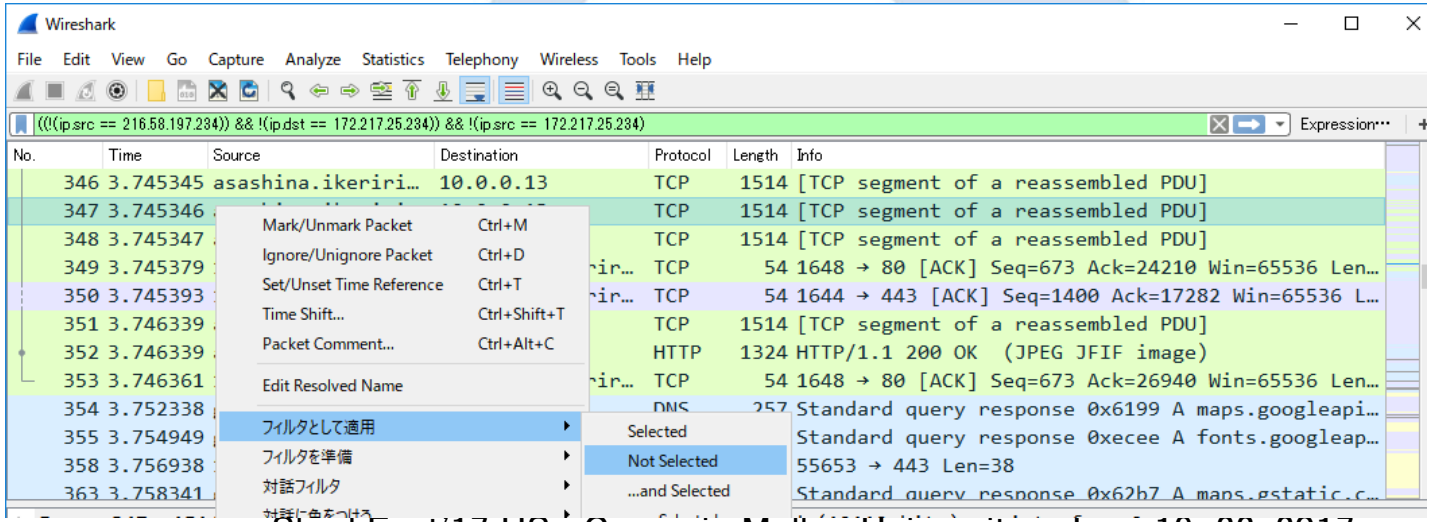
#1. Collect only troubled IEEE802.11 packet
Excercise 1

You can collect Deauthentication/Disassociation,
weather packet is encrypted or plaintext
(because management frames are sent in clear text)
Using capture filter is also good for reducing pcapng size.



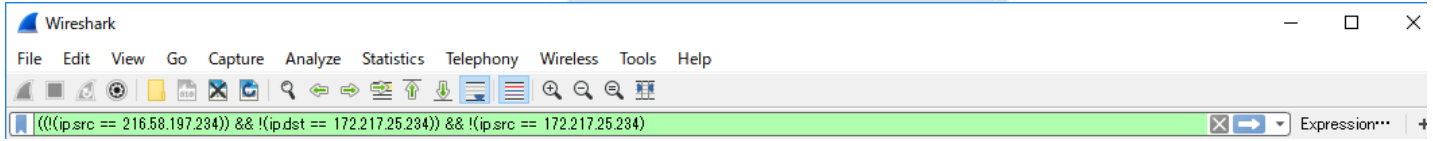
#2.Use the exclude Display filter

If you have no idea to find packet you needed, Use the exclude Display filter is the easiest and best way, and do not forget right click and choose **Apply filter ＞ „... and not selected „**



#2.Use the exclude Display filter

- The direction („ip.src“ or „ip.dst“) is determined by **the position of right click** in packet list pane.
- Applying exclude display filter many times...
  ((!(ip.src == 216.58.197.234)) && !(ip.dst == 172.217.25.234)) && !(ip.src == 172.217.25.234)...
- You may find the packets you have not recognized.
- Excluding MAC Address is also useable

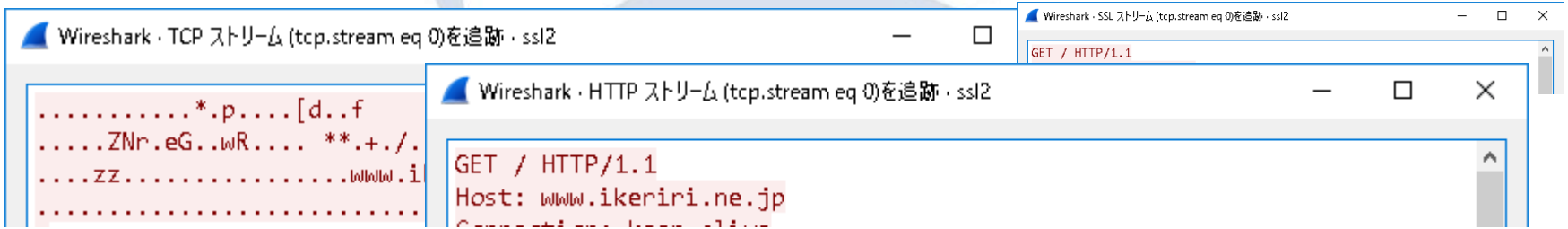


#2.Use the exclude Display filter
Excercise 2

1. Open wireless.pcapng file
2. Choose „Broadcast“ in Destination row in packet list pane, right click, apply filter, „not selected“
3. Choose „02:cc:37:11:8a:04“ in Destination row, right click, apply filter, ... „and not selected“
4. set WPA-PWD (pass phrase : SSID)
   sharkfest:sharkfest17-ikeriri in Decryption Keys...
   IEEE802.11 wireless LAN preferences,
   Then we got the packets we wanted.

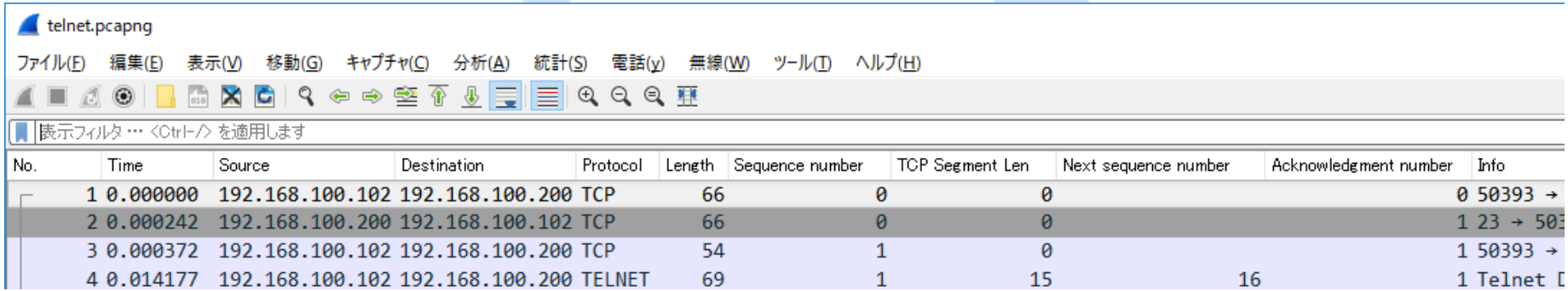
#3 Stream ID and follow TCP in 2 way

- Wireshark set sequence number of transaction of TCP, UDP, SSL and HTTP. For example Wireshark find first SYN packet, and set stream number.
- We can check flow using straem id using display filter. tcp.stream udp.stream ssl.stream http.stream
- It is very useful to grab TCP/UDP/SSL stream



#3 Stream ID and follow TCP in 2 way

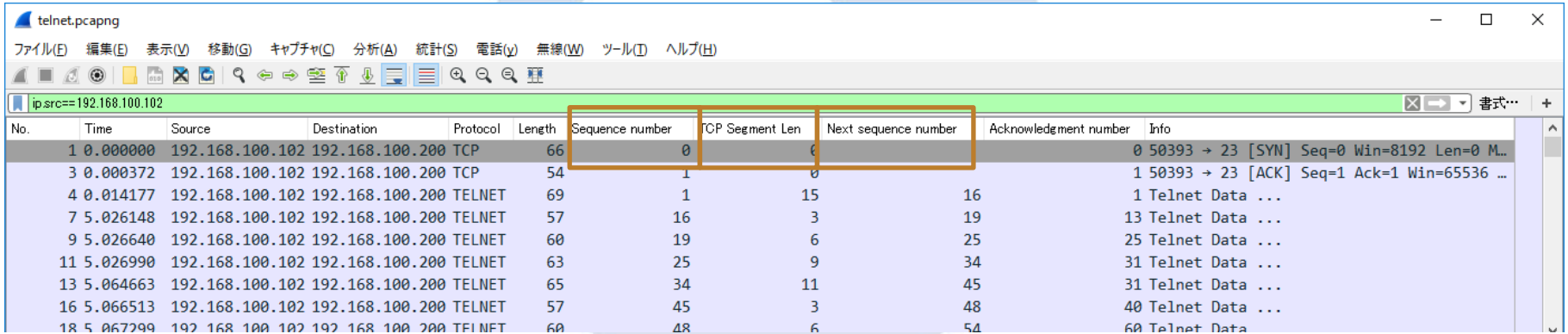
- Let‘s add columns
  Sequence no (tcp.seq)
  TCP Segment Len (tcp.len)
  Next sequence no (tcp.nxtseq)
  Acknowledgement no (tcp.ack)



No.	Time	Source	Destination	Protocol	Length	Sequence number	TCP Segment Len	Next sequence number	Acknowledgement number	Info
1	0.000000	192.168.100.102	192.168.100.200	TCP	66	0	0		0	50393 →
2	0.000242	192.168.100.200	192.168.100.102	TCP	66	0	0		1	23 → 503
3	0.000372	192.168.100.102	192.168.100.200	TCP	54	1	0		1	50393 →
4	0.014177	192.168.100.102	192.168.100.200	TELNET	69	1	15	16	1	Telnet D

#3 Stream ID and follow TCP in 2 way

- set ip.src== client/server IP address in display filter, so you can understand the mechanizm of TCP
  (seq no + segment length = next sequence number)

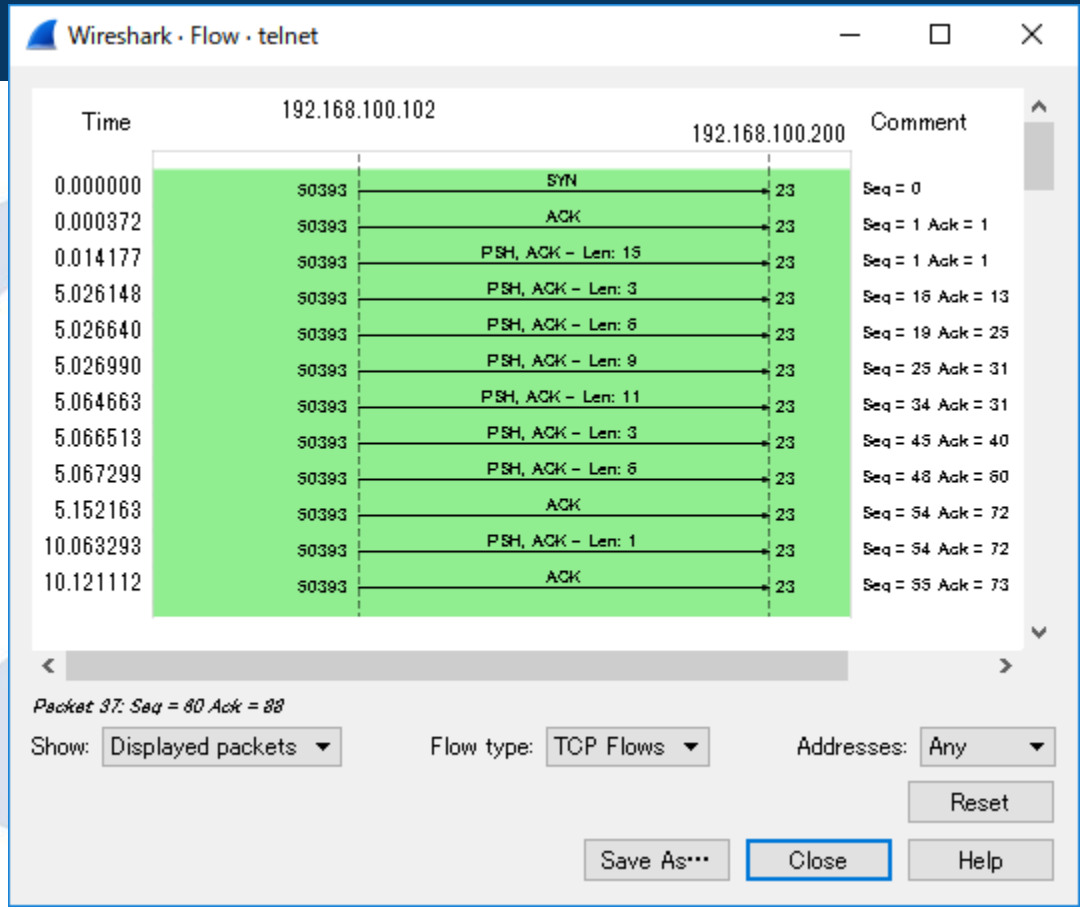


No.	Time	Source	Destination	Protocol	Length	Sequence number	TCP Segment Len	Next sequence number	Acknowledgment number	Info
1	0.000000	192.168.100.102	192.168.100.200	TCP	66	0	0		0	50393 → 23 [SYN] Seq=0 Win=8192 Len=0 M…
3	0.000372	192.168.100.102	192.168.100.200	TCP	54	1	0		1	50393 → 23 [ACK] Seq=1 Ack=1 Win=65536 …
4	0.014177	192.168.100.102	192.168.100.200	TELNET	69	1	15	16	1	Telnet Data ...
7	5.026148	192.168.100.102	192.168.100.200	TELNET	57	16	3	19	13	Telnet Data ...
9	5.026640	192.168.100.102	192.168.100.200	TELNET	60	19	6	25	25	Telnet Data ...
11	5.026990	192.168.100.102	192.168.100.200	TELNET	63	25	9	34	31	Telnet Data ...
13	5.064663	192.168.100.102	192.168.100.200	TELNET	65	34	11	45	31	Telnet Data ...
16	5.066513	192.168.100.102	192.168.100.200	TELNET	57	45	3	48	40	Telnet Data ...
18	5.067299	192.168.100.102	192.168.100.200	TELNET	60	48	6	54	60	Telnet Data

#3 Stream ID and follow TCP in 2 way

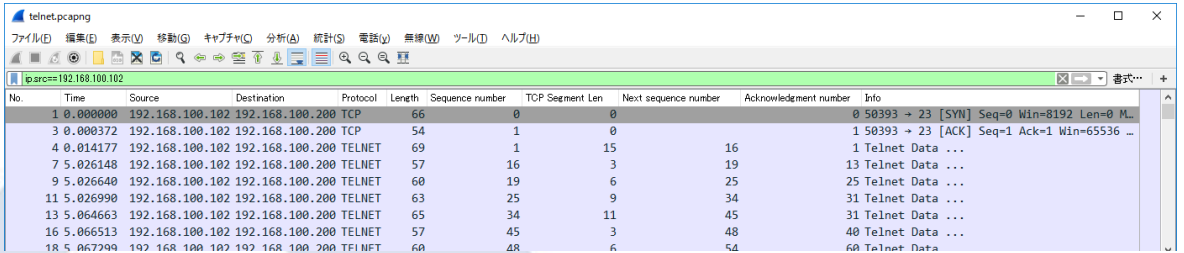
Use flow graph
(Flow type as TCP)

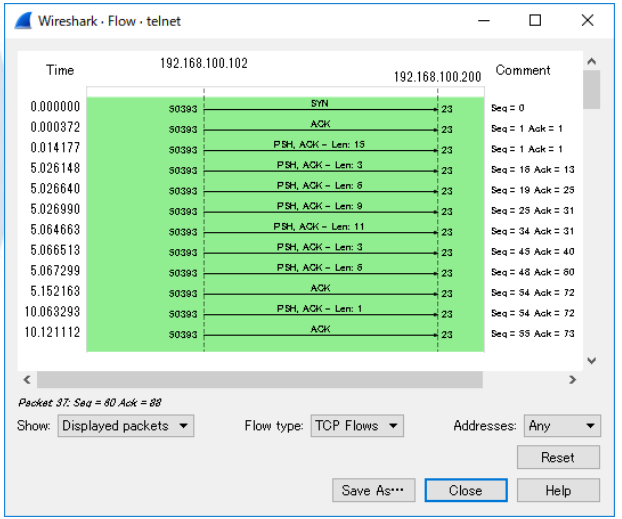
- It is another way to understand the flow of TCP from a peer.
- Statistics > Flow Graph, choose Displayed packets in Show, choose TCP flow in flow type



#3 Stream ID and follow TCP in 2 way
Excercise 3

- Open telnet.pcapng
- Set as columns of
  Sequence no (tcp.seq)
  TCP Segment Len (tcp.len)
  Next sequence no (tcp.nxtseq)
  Acknowledgement no (tcp.ack)
- Set display filter as ip.src
  from Client or Server IP address
- Choose Statistics > Flow Graph,
  choose Displayed packets in Show,
  choose TCP flow in flow type





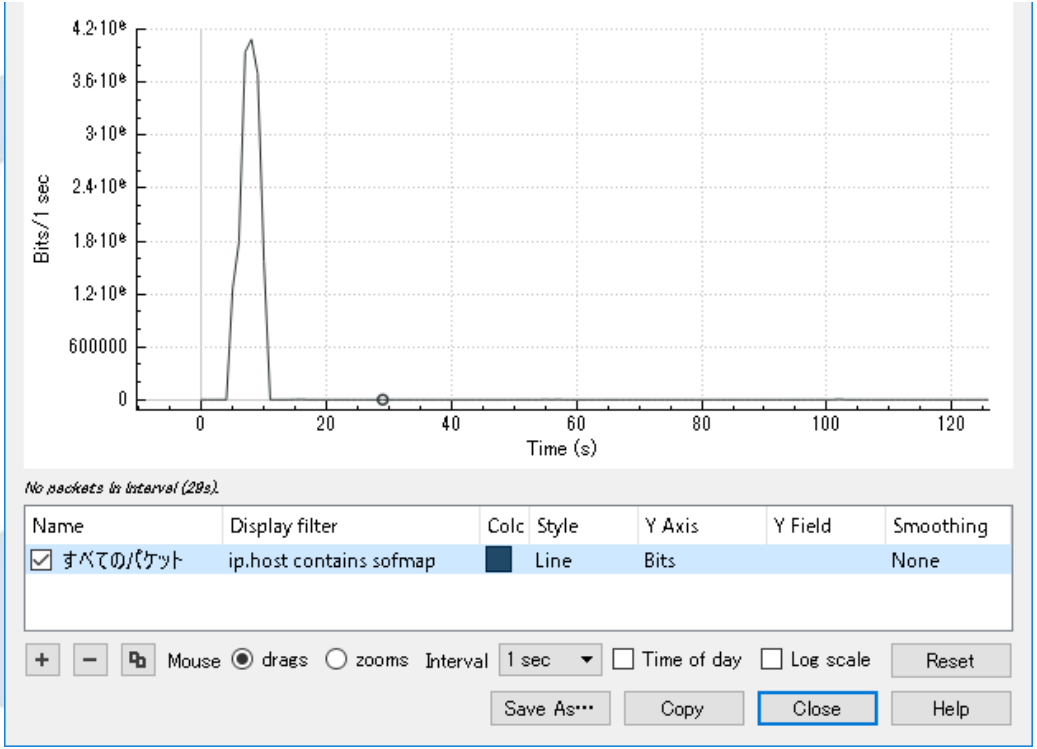
#4 Visualize using I/O Graph
New Wireshark I/O Graph is very much useful

I/O Graph (set Y axis as bits, packets, and Y fields)		
Bits (line)	packets (histogram)	Y fields
bits/s and bytes/s	Frequency	Time, Size, …

- To see performance and throughput,
  set Y axis to bit/sec (style:line)
- To see errors and counting the number of packet,
  set Y axis to packet/sec (style:histogram)
- To see time or size or other field value,
  set Y axis to scholar function of Y fields

#4 Visualize using I/O Graph
4-1 To see performance and throughput

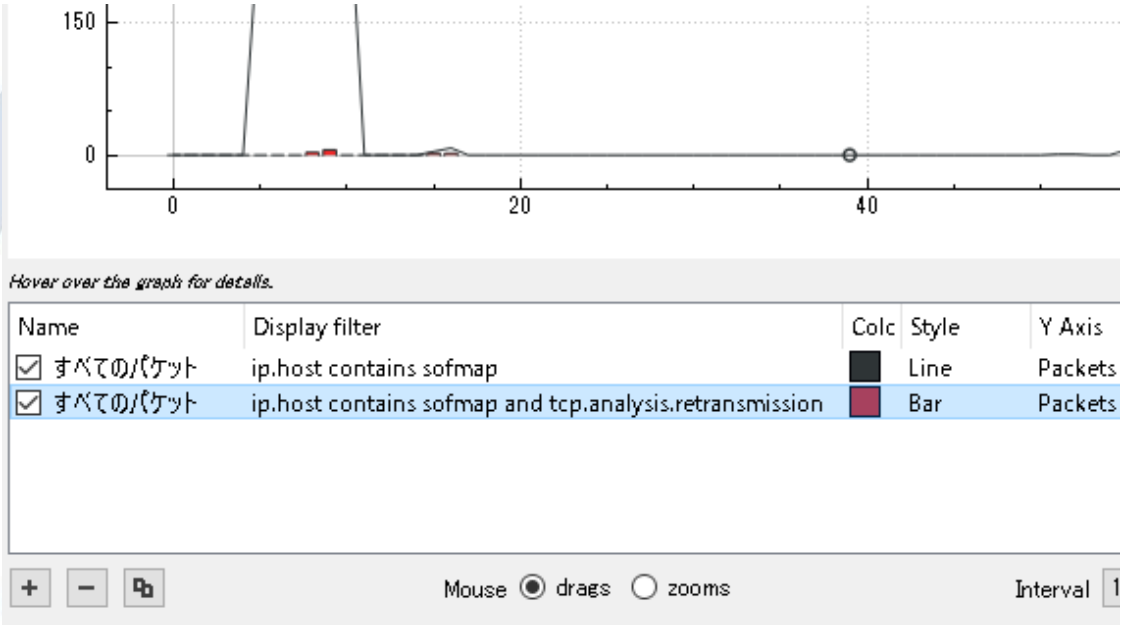
1. Add Item
2. Set Display filter as ip.host contains FQDN (sofmap.com)
3. Set Style as Line
4. set Y axis to bit/sec
5. Check Interval 1 sec



#4 Visualize using I/O Graph
4-2 To see errors and counting frame

webbrowse.pcang

1. Copy current item
2. Add the item's Display filter as "... tcp.analysis. retransmission"
3. Set Style as Bar
4. set Y axis to Packets/sec



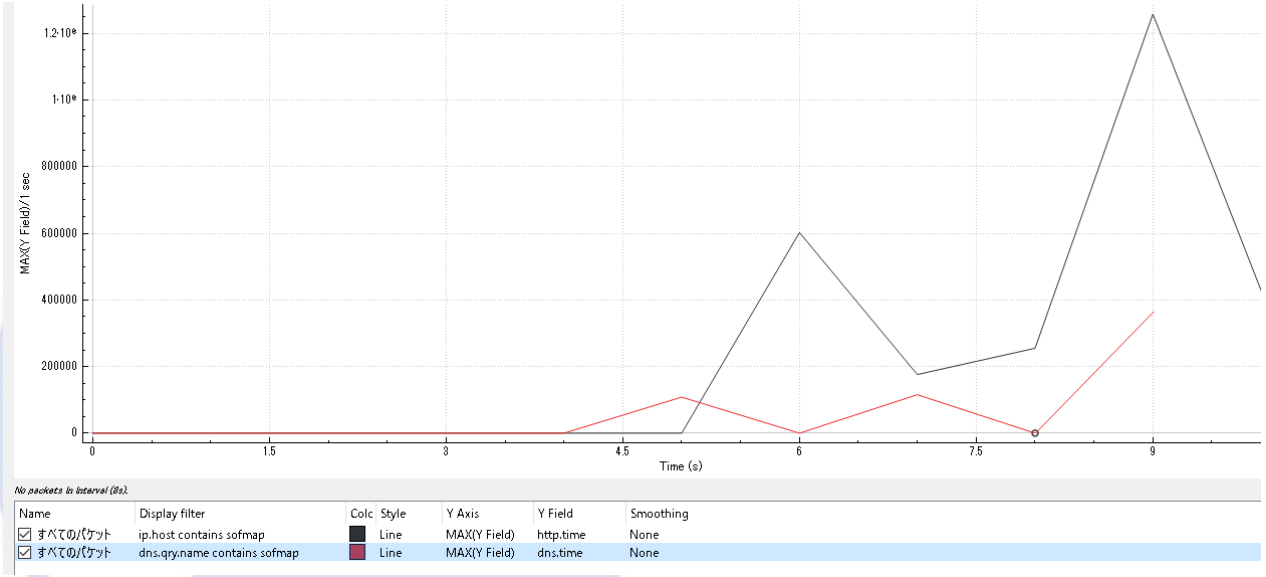
#4 Visualize using I/O Graph

4-3 To see time or size or other field value,

webbrowse.pcang

1. Delete 2nd item
2. Set Y Axis as “MAX(Y Field)”
3. Set Y field as “http.time”
4. Copy item
5. Set Display filter as “dns.qry.name contains FQDN”
6. Set Y field as “dns.time”

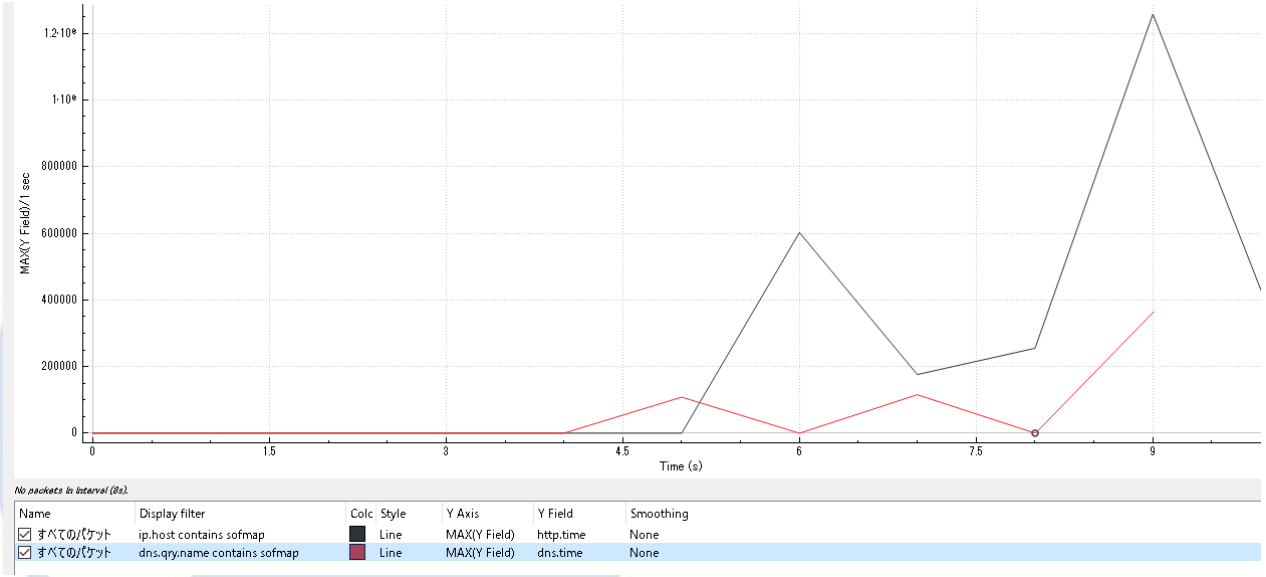
compare



#4 Visualize using I/O Graph

4-3 To see time or size or other value, webbrowse.pcang

1. Delete 2nd item
2. Set Y Axis as “MAX(Y Field)”
3. Set Y field as “http.time”
4. Copy item
5. Set Display filter as “dns.qry.name contains FQDN”
6. Set Y field as “dns.time”



#4 Visualize using I/O Graph
Excercise 4

- To see performance and throughput
  Set Y axis as bits
- To see errors and counting
- Set Y axis as packets
- To see time or size or other value,
  Set Y axis as scholar function
  Set Y field value as time or size

#5 Capture packet periodically and merge for stats.

- dumpcap/tshark/wireshark has the same CLI option to capture packets for long term,
- At first check the interface index with -D option

dumpcap / tshark / wireshark -b (capture multiple files)		
duration:NUM	filesize:NUM	files:NUM
switch to next file after NUM secs	switch to next file after NUM KB	ringbuffer: replace after NUM files

- The pcap files are created as name_5digit-seq_YYYYMMDDHHMMSS.pcapng

#5 Capture packet periodically and merge for stats.

- Using duration option is good for statistics.
- You can set duration as you want to set Units for stats.
  60(1 minutes) / 3600(1 hour) / 86400(1 day)
- –a (autostop) option is available for large size statistics

dumpcap / tshark / wireshark –a (autostop option)		
duration:NUM	filesize:NUM	files:NUM
Stop after NUM secs	Stop after NUM KB	Stop after NUM files

- After capturing, you just check the number of files, then merge using wildcard option of mergecap.

#5 Capture packet periodically and merge for stats.

```
C:¥Users¥megumi¥Desktop>dumpcap -D
1. ¥Device¥NPF_{2D82BEBC-1E21-4E44-AF3E-3F9C4EA6748C} (繝ｭ繝ｼ繧ｫ繝ｫ 繧ｨ繝ｪ繧｢謗･
2. ¥Device¥NPF_{4F0D27B0-E075-4FF7-9D28-BA58ABD38307} (繝ｭ繝ｼ繧ｫ繝ｫ 繧ｨ繝ｪ繧｢謗･
3. ¥Device¥NPF_{F4FFD736-AD7B-455A-9DD7-EAF47FB51F31} (ethernet)
4. ¥Device¥NPF_{B98C1496-712F-4755-AEB9-87981430B800} (local area connection)
5. ¥Device¥NPF_{411DFF04-625C-4BA3-A3DB-25EFF691C708} (Wi-Fi)

C:¥Users¥megumi¥Desktop>dumpcap -i 3 -w stat.pcapng -b duration:60 -a files:60
Capturing on 'ethernet'
File: stat_00001_20170530143410.pcapng
```

- dumpcap –b duration:60 –a files:60 means we capture 60 files (1 hour) that each duration is 60 seconds (1 minute)
- You can merge packets for statistics, just counting files.
- For example, you can merge these files just using wildcard

```
C:¥Users¥megumi¥Desktop>mergecap * -w 1hourstat.pcapng
```

Includes Stat_00001_…～
Stat_00061_…(60 files)

#5 Capture packet periodically and merge for stats. Exercise 5

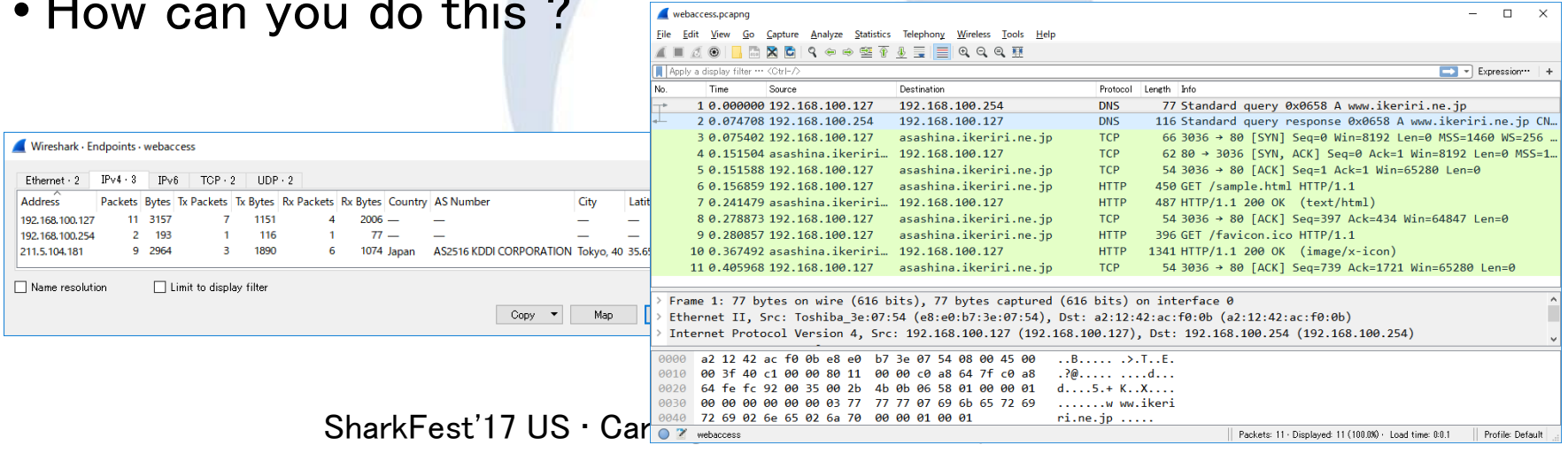
- Check interface index using –D option
- Set 6 as duration with –b, set 10 as files with –a option
- Capture packet to 10 test.pcapng files each duration is 6 second

```
C:¥Users¥megumi¥Desktop>dumpcap -i 1 -b duration:6 -a files:10 -w test.pcapng
Capturing on '縺ｭ縺ｼ繧ｵ縺ｮ 繧ｲ縺ｭ繧「謗･邯・ 11'
File: test_00001_20170531165817.pcapng
```

- Merge 10 files (60 seconds) using wildcard
- Create stats using tshark –qz option.
  conv,ip conv,wlan dns,tree endpoints,ip endpoints,tcp endpoints,wlan expert follow,http hosts http2,tree http_req,tree http_srv,tree io,phs io,stat

#6 Sort and stack the pcap file automatically and divide by each filter value

- There are tons of bored and time-wasting pcapng task when we inspect and dissect packets using Wireshark
- Here is a very simple web access pcap file that contains 3 hosts.
- You need to create pcap files filtered with each host.
- How can you do this ?

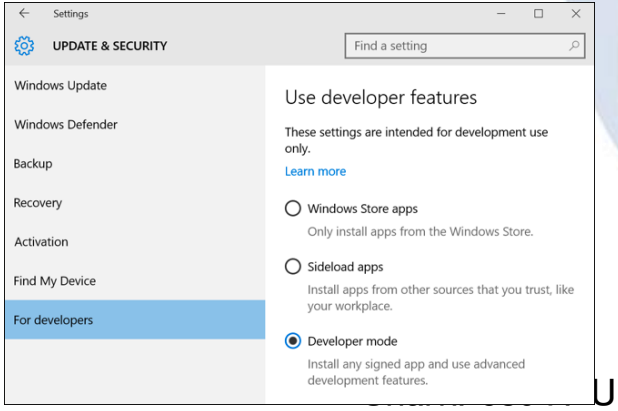


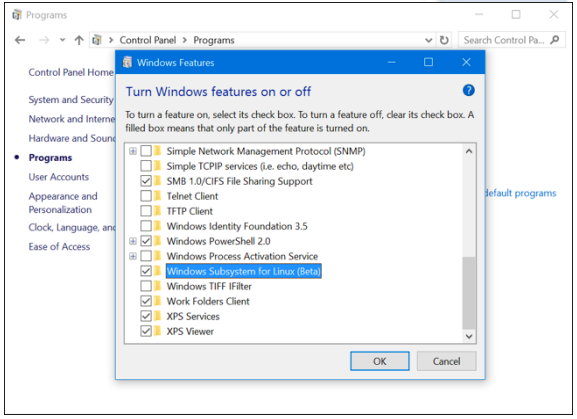
#6 Sort and stack the pcap file automatically and divide by each filter value

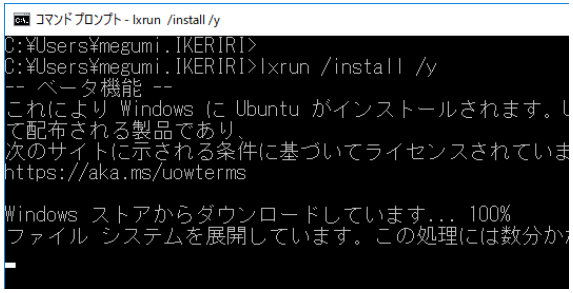
- Windows10 64bit version supports bash shell
- You can add bash and use another convenient shell command
  You can extend wireshark CUI command with these shell script and process packet task automatically.
- Please note do not think just 1 command line,
  but you can do task with creating temporal.file
- (1) Collect source host list from the pcapng files
  Hint: tshark -T fileds and -e option,
  (2) Filter packets by source host
  and create each filtered pcapng file.
  Hint: tshark -Y ip.src and shell script (for)

#6 Sort and stack the pcap file automatically and divide by each filter value

- Open Settings > Update & Security > For Developers and activate the Developer Mode
- Open Control Panel > Adding and Deleting Programs > Turn Windows Features On or Off and check Windows Subsystem for Linux (Beta)
- Open cmd.exe and type bash, or lxrun /install /y







#6 Sort and stack the pcap file automatically
(1) Collect source host list from the pcapng files

- Wireshark display filter “ip.src_host” means source IP host
- You can pick up source host list using “Tshark -T fields -e ip.src_host”

```
C:¥Users¥megumi¥Desktop>tshark -r web
access.pcapng T fields -e ip.src_host

192.168.100.127
                GW
192.168.100.127
asashina.ikeriri.ne.jp
192.168.100.127
192.168.100.127
asashina.ikeriri.ne.jp
192.168.100.127
192.168.100.127
asashina.ikeriri.ne.jp
192.168.100.127
```

tshark returns output stream with duplication

#6 Sort and stack the pcap file automatically
(1) Collect source host list from the pcapng files

- You can sort output stream using sort command at first, then delete duplication using uniq command

```
C:¥Users¥megumi¥Desktop>bash
user@LetsNoteJ10:/mnt/c/Users/megumi/
Desktop$ /mnt/c/"Program Files"/Wires
hark/tshark.exe -r webaccess.pcapng -
T fields -e ip.src_host | sort
192.168.100.127
192.168.100.127
192.168.100.127
192.168.100.127
192.168.100.127
192.168.100.127
192.168.100.127
GW
asashina.ikeriri.ne.jp
asashina.ikeriri.ne.jp
asashina.ikeriri.ne.jp
user@LetsNoteJ10:/mnt/c/Users/megumi/
Desktop$
```

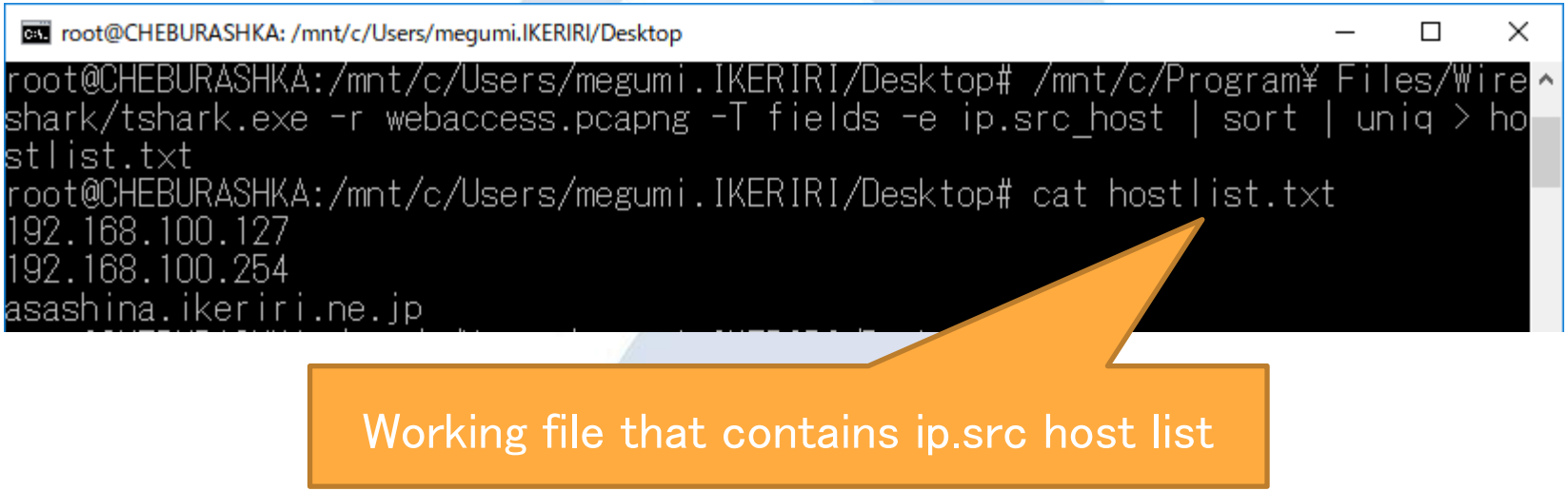
Sort with duplication at first

```
user@LetsNoteJ10:/mnt/c/Users/megumi/
Desktop$ /mnt/c/"Program Files"/Wires
hark/tshark.exe -r webaccess.pcapng -
T fields -e ip.src_host | sort | uniq
192.168.100.127
GW
asashina.ikeriri.ne.jp
```

Delete duplication with uniq command

#6 Sort and stack the pcap file automatically
(1) Collect source host list from the pcapng files

- Piping “| sort | uniq” give a list without duplication
- You can create temporal working file (hostlist.txt) using redirect at first, then delete duplication using uniq command



#6 Sort and stack the pcap file automatically
(2) Filter packets by source host and create file

- Filter packets by source host and create each filtered pcapng file.

```
C:¥Users¥megumi¥Desktop>tshark -r webaccess.pcapng -Y ip.src_host==192.168.100.127
 -w filtered_192.168.100.127.pcapng
```

- The command means open webaccess.pcapng and apply display filter “ip.src_host==192.168.100.127” then create a file
- Let’s create batch jobs for this task
- We can use “for” and use hostlist.txt
  for /f ″delims=″ %a in (hostlist.txt) do (TASK)

delimiter (¥n return code)

Parameter that contains a line that was read from hostlist.txt

#6 Sort and stack the pcap file automatically
(2) Filter packets by source host and create file

- Exit bash shell and change directory to Desktop
  test “for /f ″delims=″ %a in (hostlist.txt) do (TASK)”

```
C:¥Users¥megumi.IKERIRI>cd Desktop

C:¥Users¥megumi.IKERIRI¥Desktop>for /f "delims=" %a in (hostlist.txt) do (echo %a)

C:¥Users¥megumi.IKERIRI¥Desktop>(echo 192.168.100.127 )
192.168.100.127

C:¥Users¥megumi.IKERIRI¥Desktop>(echo 192.168.100.254 )
192.168.100.254

C:¥Users¥megumi.IKERIRI¥Desktop>(echo asashina.ikeriri.ne.jp )
asashina.ikeriri.ne.jp
```

Delimiter (¥n return code)

Display a line that was read from hostlist.txt

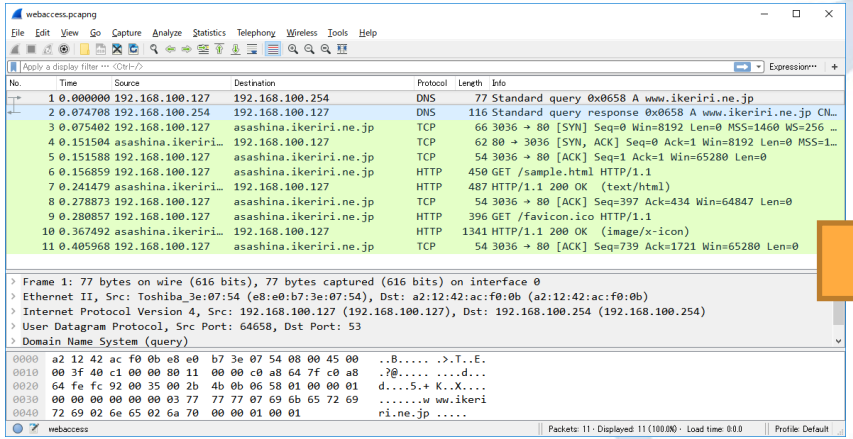
#6 Sort and stack the pcap file automatically Exercise 6

- Collect source ip host list from webaccess.pcapng
  tshark.exe -r webaccess.pcapng -T fields -e ip.src_host | sort | uniq > hostlist.txt

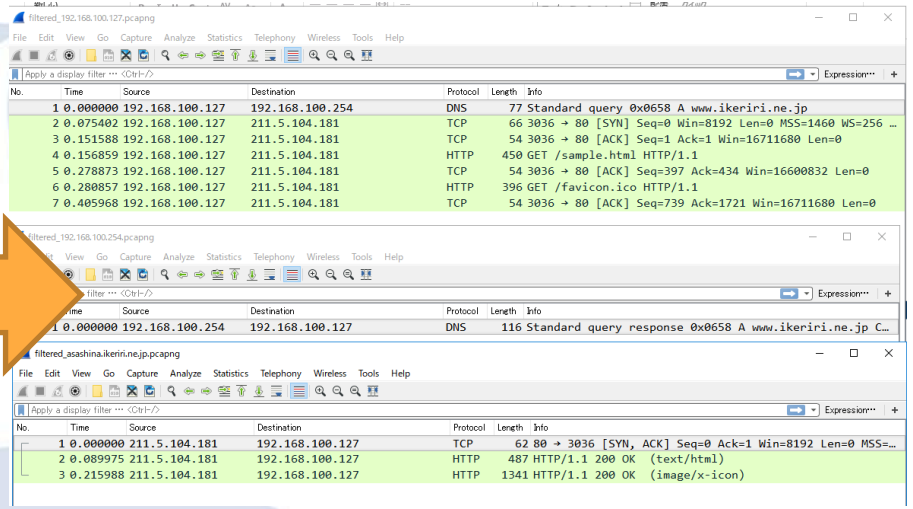
- Filter packets by source host and create pcap file that contains the each host
  for /f "delims=" %a in (hostlist.txt) do (tshark -r webaccess.pcapng -Y ip.src_host==%a -w filtered_%a.pcapng)

- Let's Combine 2STEPS

#6 Sort and stack the pcap file automatically
Compare CLI with GUI



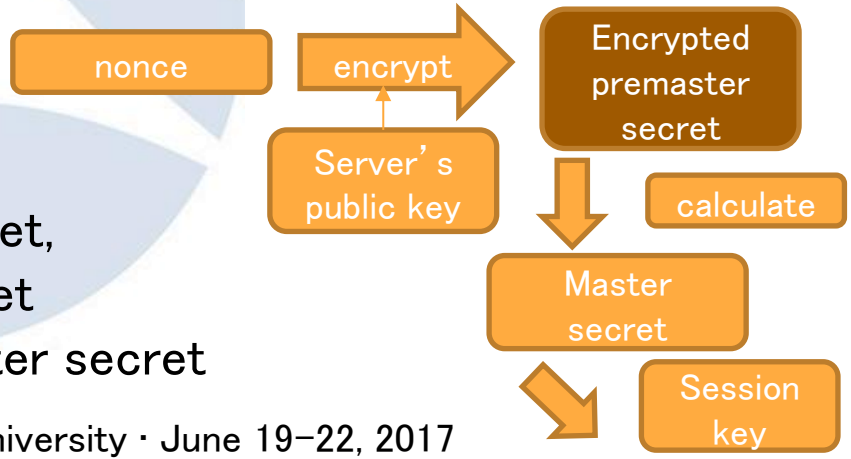
GUI



CLI

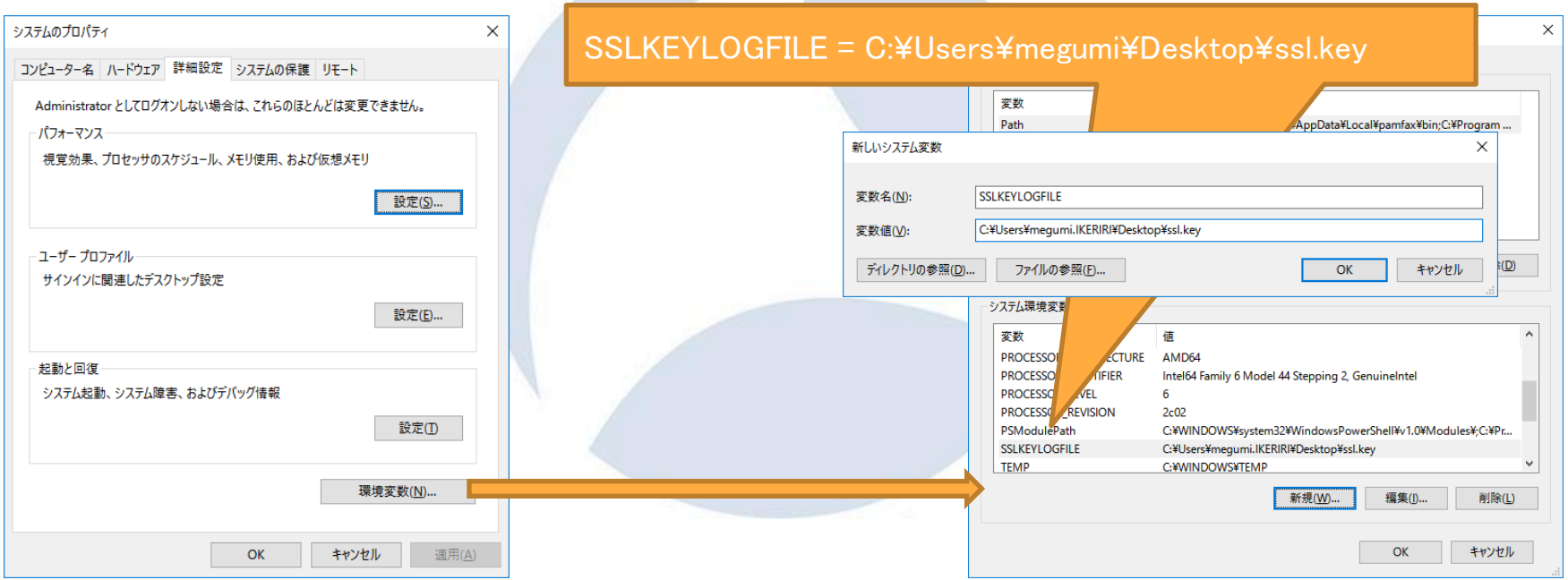
#7 Decrypting TLS/SSL data without key pair

- There are many encrypted communication using TLS/SSL in today's Internet.
- For example, HTTP2 is based on TLS/SSL
- If you have a server's private key with certification (PEM format), you can decrypt TLS/SSL trace file, but it is mere.
- If you have unencrypted premaster secret, you can retrieve session key of the SSL/TLS session
- Client creates nonce (random) and encrypt nonce by server's public key, it is encrypted premaster secret
- Master secret is created by premaster secret, then session key is created by master secret
- You need to leak your unencrypted premaster secret from Chrome.



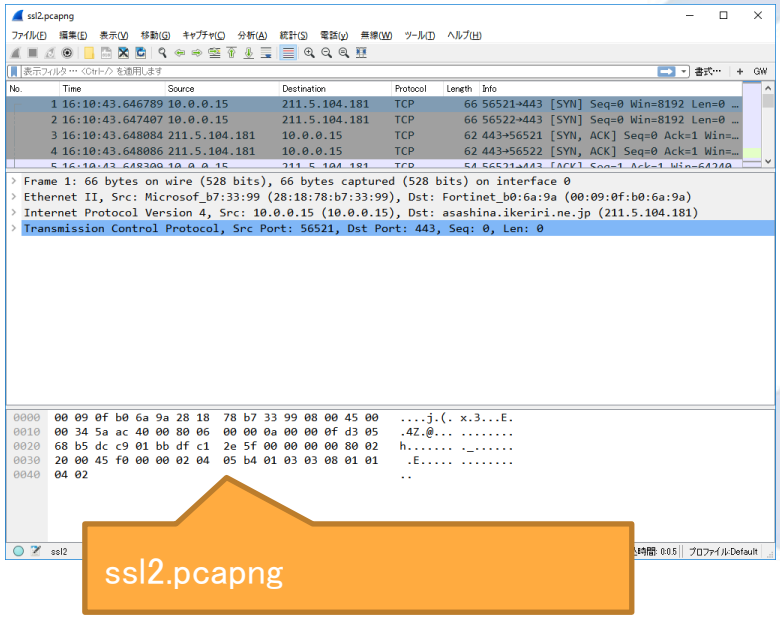
#7 Decrypting TLS/SSL data without key pair

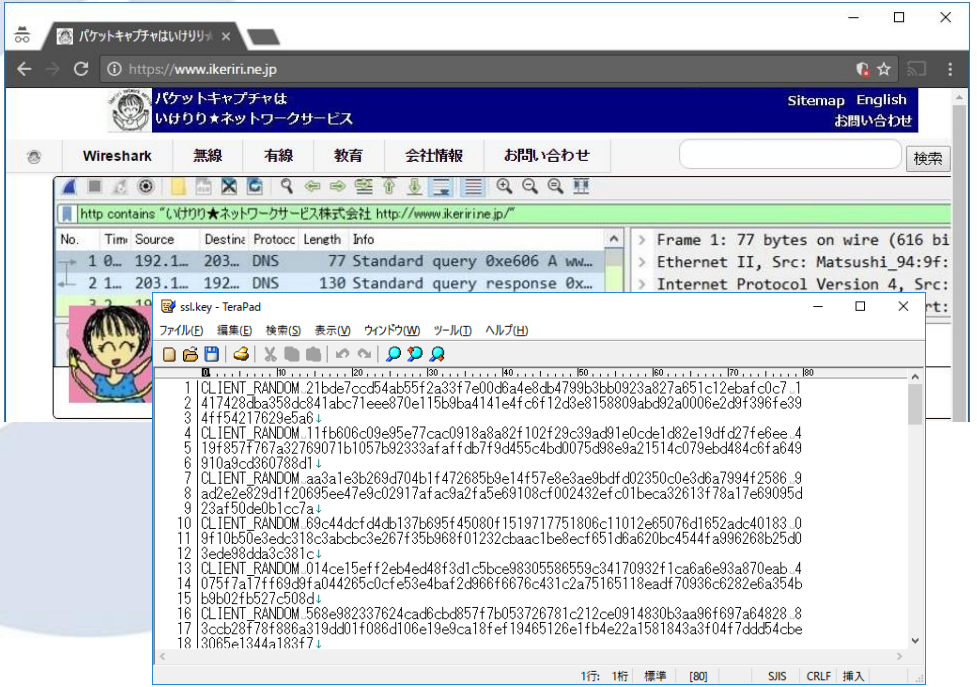
1. Set system environment variable
   SSLKEYLOGFILE=Path of the premaster secret



#7 Decrypting TLS/SSL data without key pair

2. Capture packets using Chrome (https://www.ikeriri.ne.jp/) ssl2.pcapng
   1. Check SSLKEYLOGFILE was generated

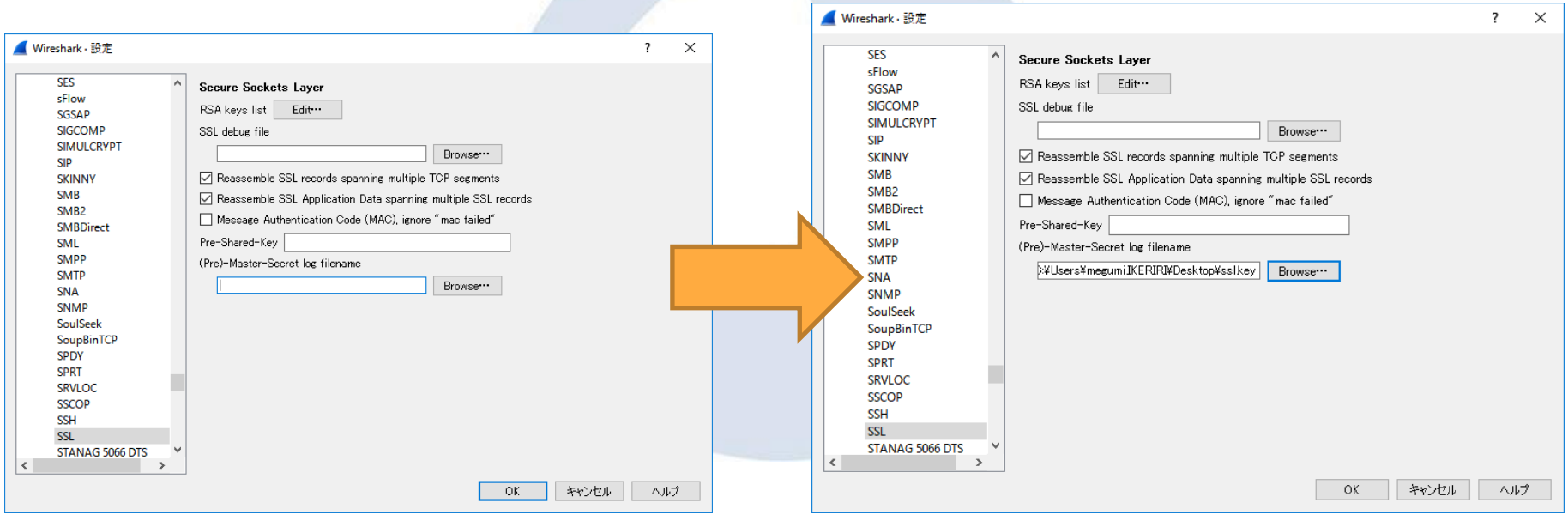




#7 Decrypting TLS/SSL data without key pair

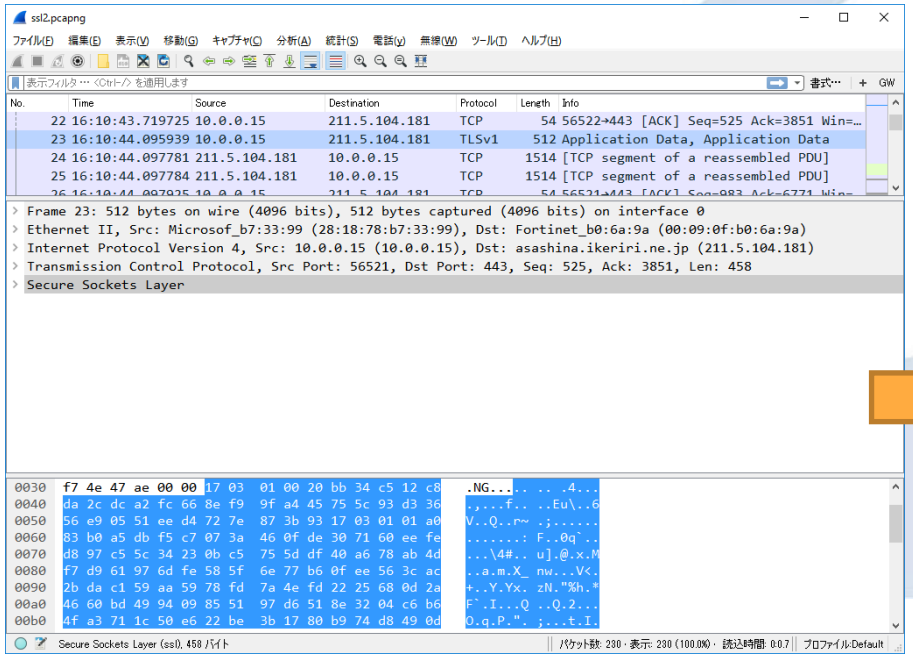
3. Set (Pre)-Master-Secret log filename in SSL preference

Add path of ssl.key in (Pre)-Master-Secret log filename



#7 Decrypting TLS/SSL data without key pair
Exercise 7

Check #23,28 in trace file ssl2.pcapng



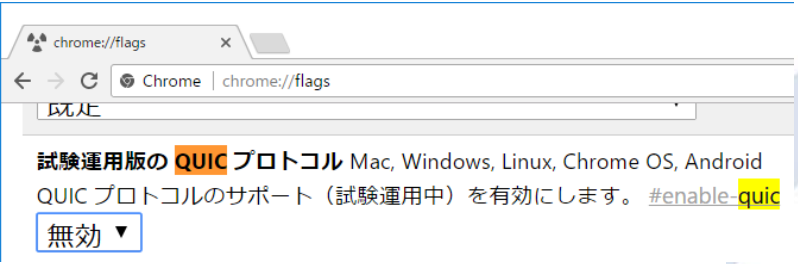
#8 HTTP2 dissection

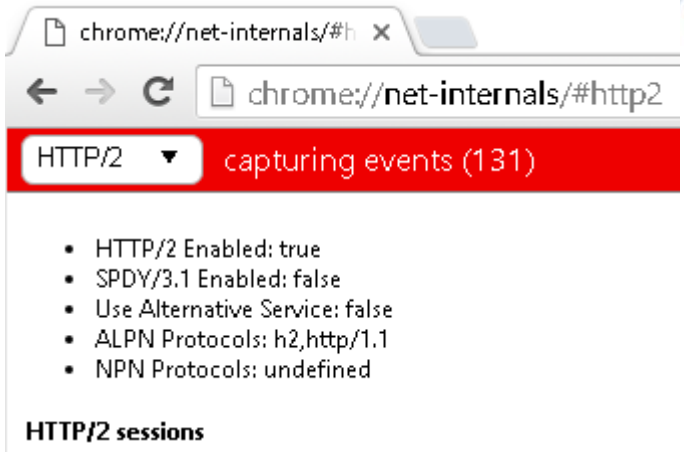
- In Windows10 age, major browser and websites are ready for HTTP2 for example Google Map and Gmail, Facebook, Twitter, Yahoo, IE, Chrome, Safari….
- HTTP/2.0 uses with TLS and all traffic is encrypted.
  So we need to use Wireshark to decrypt them

http2

No.	Tim	Source	Destination	Protocol	Length	Info
	0…	23.78.…	192.168.0.…	HTTP2	135	SETTINGS, WINDOW_UPDATE
	0…	192.16…	23.78.84.1…	HTTP2	153	Magic, SETTINGS, WINDOW_UPDATE
	0…	192.16…	23.78.84.1…	HTTP2	478	HEADERS, WINDOW_UPDATE
	0…	192.16…	23.78.84.1…	HTTP2	104	SETTINGS
	0…	23.78.…	192.168.0.…	HTTP2	104	SETTINGS
	0…	23.78.…	192.168.0.…	HTTP2	1286	HEADERS, DATA, DATA
	0…	23.78.…	192.168.0.…	HTTP2	104	DATA

#8 HTTP2 dissection
HTTP/2.0 uses SSL/TLS

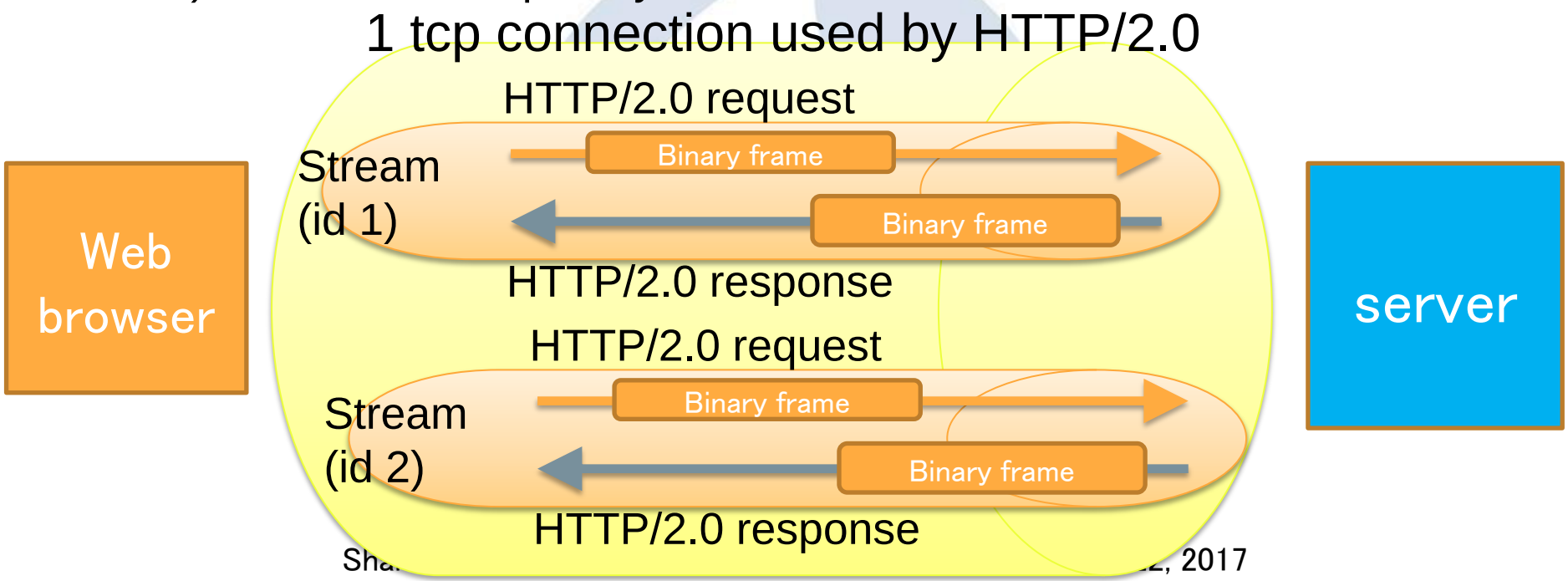




- Open Chrome URL “chrome://flags/”
- Set Disable to QUIC protocol in list box
- Start capture and open www.twitter.com in Chrome and save twitter.pcapng
- Open the Chrome and type chrome://net-internals/#http2 you can see the HTTP/2 sessions
- Wireshark has the dissector of HTTP2.0 https://www.wireshark.org/docs/dfref/h/http2

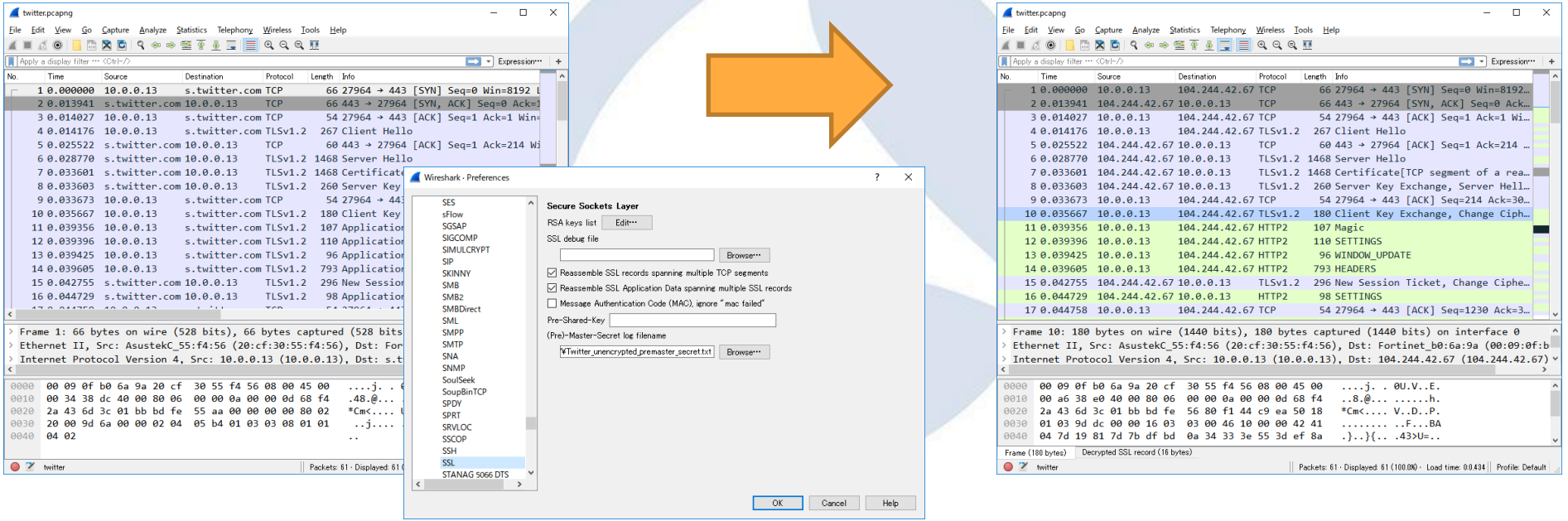
#8 HTTP2 dissection
Stream ID is important to dissect HTTP2

HTTP/2.0 manages communications using Stream mechanism
HTTP/2.0 uses 1 tcp connection and many Stream (virtual connection channel) that has id and priority



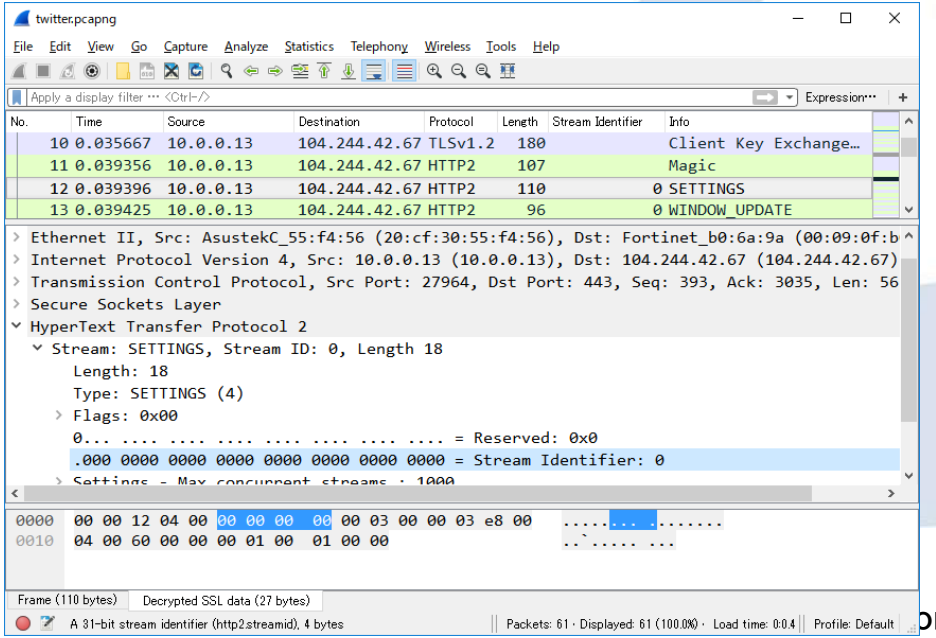
#8 HTTP2 dissection
Exercise 8 Decryption setting of SSL/TLS

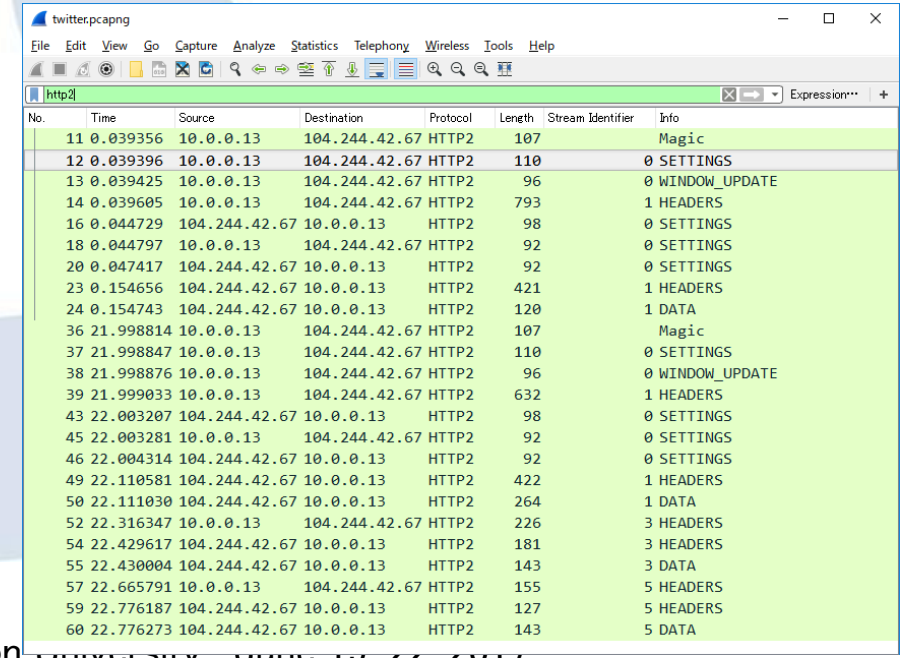
Open twitter.pcapng and set (Pre)-Master-Secret log filename Twitter_unencrypted_premaster_secret in SSL preference



#8 HTTP2 dissection
Exercise 8 Add stream ID row and filter http2

Select packet 1 and open HTTP2 header, and select Stream Identifier (http2.streamid) and right click and select “apply as column”
Set http2 in Display Filter

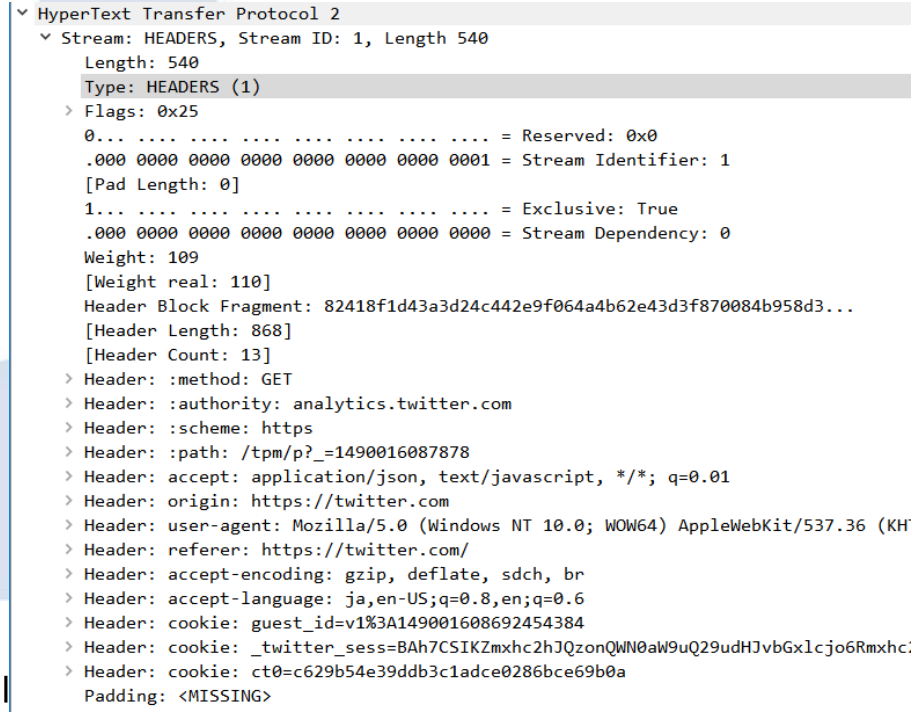




#8 HTTP2 dissection
Exercise 8 check HTTP2 request header

- Set http2.header.value == "GET“ (it means HEADER(http2.type == 1))
- Check 39 packet and open HTTP2 header

- HTTP2 are binary frame
- Check major header field
- Method HTTP method
- Authority Host of the server
- Scheme HTTP / HTTPS
- Path Path of the object
- Accept accept document type
- User-agent browser information
- Referrer last visited uri
- Cookie



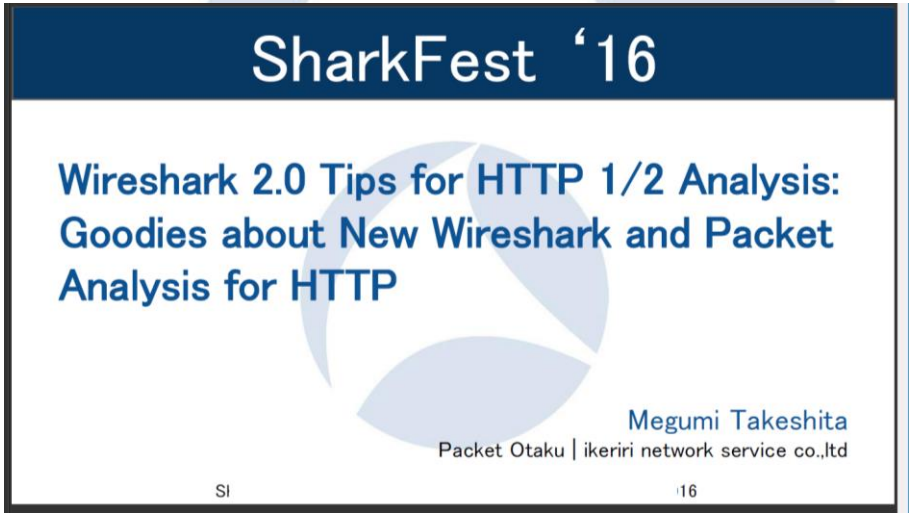
#8 HTTP2 dissection
Exercise 8 check HTTP2 response header

- http2.header.value == "200“ (it means HEADER(http2.type == 1))
- Check 49 packet and open HTTP2 header

- HTTP2 are binary frame
- Check major header field
- :status Status code
- Content-encoding compression
- Server Web server
- content-type document type

#8 HTTP2 dissection more information

For more detail information about HTTP2
Please refer my last year's presentation at Sharkfest 2016
"Wireshark 2.0 Tips for HTTP and HTTP2 Analysis"

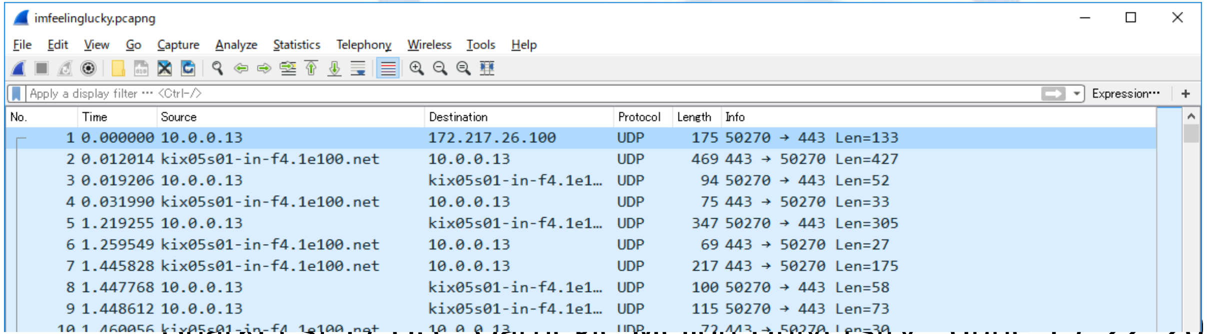


#9 QUIC dissection
What is QUIC

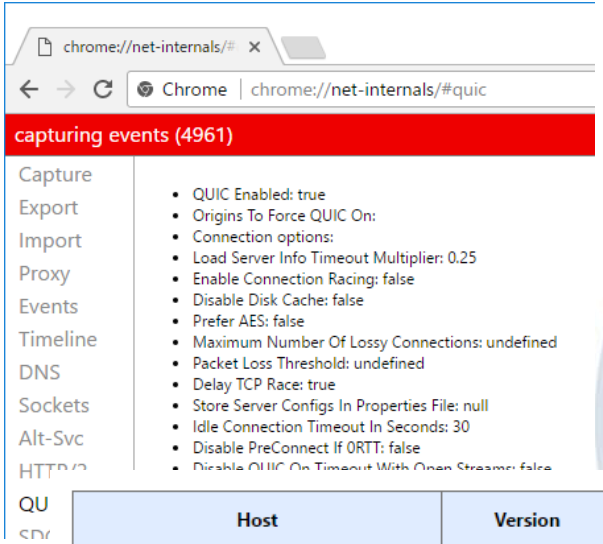
- TCP 3 way handshake needs more connection time.
  (off course there is a TCP Fast Open)
- TCP have to send new segment after old segment is confirmed as received successfully (Head of Line Blocking)
- TCP resend timer calculation and receive window mechanism are not adequate in today's internet
- Anyway, TCP is slow protocol, so SSL/TLS is slow, and HTTP is also slow, so Google create new protocol instead of TCP/(SSL/TLS)/(HTTP/HTTP2) model
- Google creates QUIC.

#9 QUIC dissection
capturing QUIC

- QUIC (Quick UDP Internet Connection) is Internet-Drafts https://datatracker.ietf.org/wg/quic/documents/ no RFC#
- open Wireshark and start capturing
  open www.google.com and I'm feeling lucky, stop capture.
  Then save filtered pcapng as imfeelinglucky.pcapng
- At this time we just see some UDP stream



#9 QUIC dissection
check QUIC connection

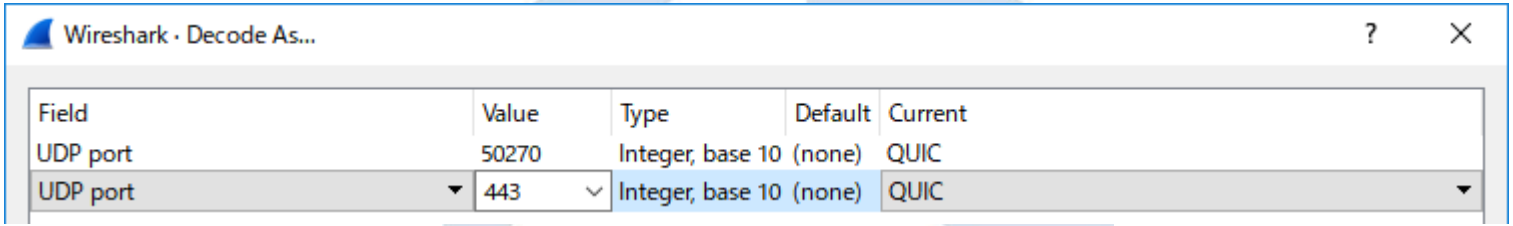


- Open the Chrome and type chrome://net-internals/#quic you can see the QUIC sessions
- Please check Connection UID

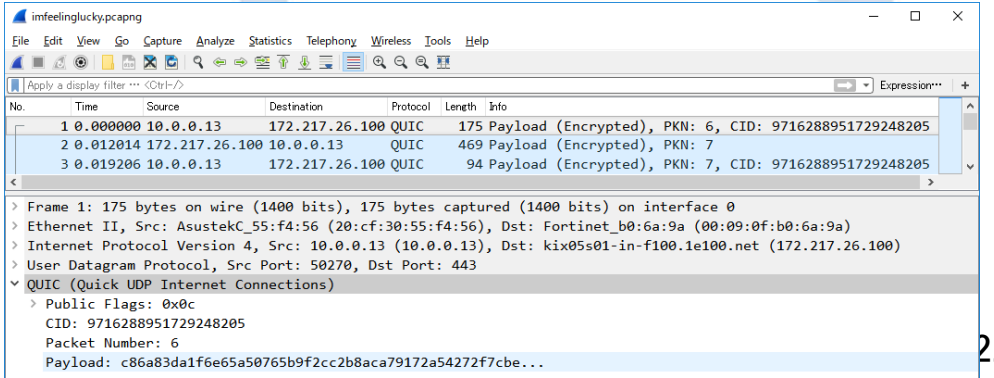
Host	Version	Peer address	Connection UID	Active stream count	Active streams	Total stream count	Packets Sent	Packets Lost	Packets Received	Connected
apis.google.com:443 ogs.google.com:443	QUIC_VERSION_35	216.58.197.142:443	8126651583460386461	0	None	1	7	0	7	true
clients4.google.com:443	QUIC_VERSION_35	172.217.26.46:443	7635153523791641412	0	None	1	8	0	7	true
clients5.google.com:443	QUIC_VERSION_35	172.217.26.46:443	3964807834305814218	0	None	0	3	0	3	true
fonts.gstatic.com:443 ssl.gstatic.com:443	QUIC_VERSION_35	172.217.26.35:443	3002020916089671232	0	None	0	3	0	3	true
lh3.googleusercontent.com:443	QUIC_VERSION_35	216.58.197.129:443	1788128156380661810	0	None	0	4	0	4	true
notifications.google.com:443 play.google.com:443	QUIC_VERSION_35	216.58.197.238:443	5902717492422574525	0	None	0	3	0	3	true
play.google.com:443	QUIC_VERSION_35	216.58.197.238:443	10382952661970550630	0	None	1	7	0	5	true
www.google.co.jp:443	QUIC_VERSION_35	216.58.197.227:443	1168159024473252919	0	None	20	70	0	97	true
www.google.com:443	QUIC_VERSION_35	172.217.26.100:443	6629889265344322331	0	None	1	5	0	4	true
www.gstatic.com:443	QUIC_VERSION_35	216.58.197.131:443	8479664014582986847	0	None	0	4	0	4	true

#9 QUIC dissection
Decode as QUIC

- Click udp packet and right click “Decode As…” and choose non-well-known port as QUIC (this time port 50270)



- Wireshark recognize the port is used for QUIC



#9 QUIC dissection
QUIC architecture

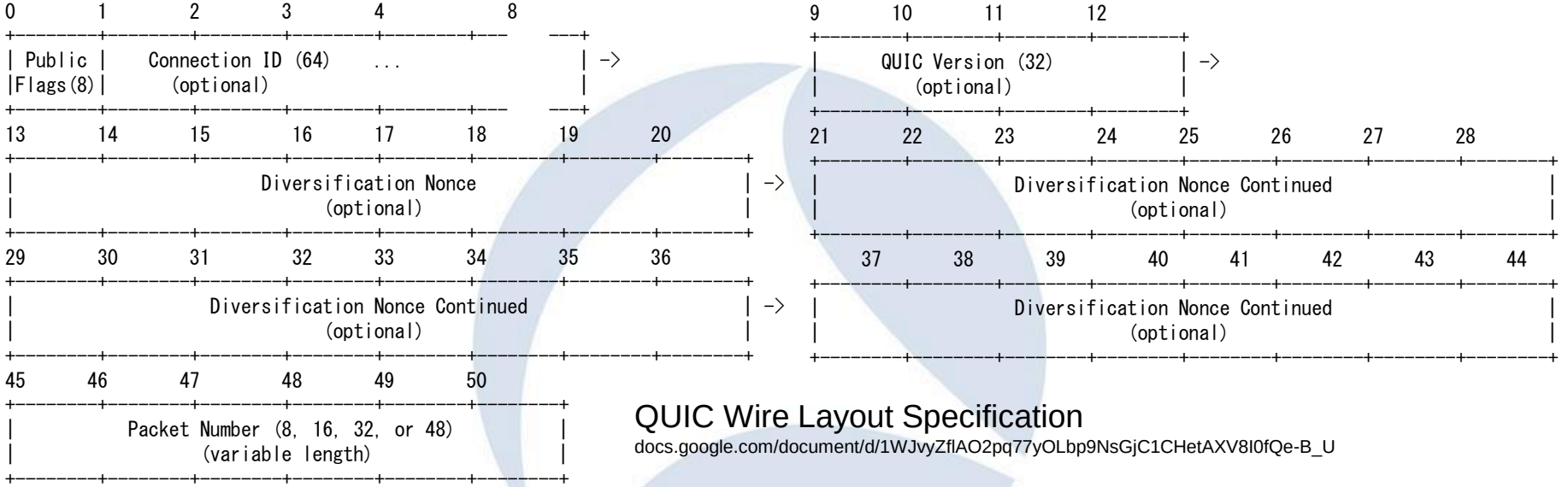
- HTTP2 (eth:ip:tcp:ssl:http2)

```
> Ethernet II, Src: AsustekC_55:f4:56 (20:cf:30:55:f4:56), Dst: Fortinet_b0:6a:9a (00:09:0f:b
> Internet Protocol Version 4, Src: 10.0.0.13 (10.0.0.13), Dst: 104.244.42.67 (104.244.42.67)
> Transmission Control Protocol, Src Port: 27964, Dst Port: 443, Seq: 393, Ack: 3035, Len: 56
> Secure Sockets Layer
v HyperText Transfer Protocol 2
```

- QUIC(eth:ip:udp:quic:http2(Encrypted))
  QUIC overlaps SSL/TLS+TCP on UDP
  with Encryption, Authentication, Session establishment,
  Flow control, Error correction and Congestion Control

```
> Frame 1: 175 bytes on wire (1400 bits), 175 bytes captured (1400 bits) on interface 0
> Ethernet II, Src: AsustekC_55:f4:56 (20:cf:30:55:f4:56), Dst: Fortinet_b0:6a:9a (00:09:0f:b0:6a:9a)
> Internet Protocol Version 4, Src: 10.0.0.13 (10.0.0.13), Dst: kix05s01-in-f100.1e100.net (172.217.26.100)
> User Datagram Protocol, Src Port: 50270, Dst Port: 443
v QUIC (Quick UDP Internet Connections)
  > Public Flags: 0x0c
    CID: 9716288951729248205
    Packet Number: 6
    Payload: c86a83da1f6e65a50765b9f2cc2b8aca79172a54272f7cbe...
```

#9 QUIC dissection
QUIC Public Packet Header

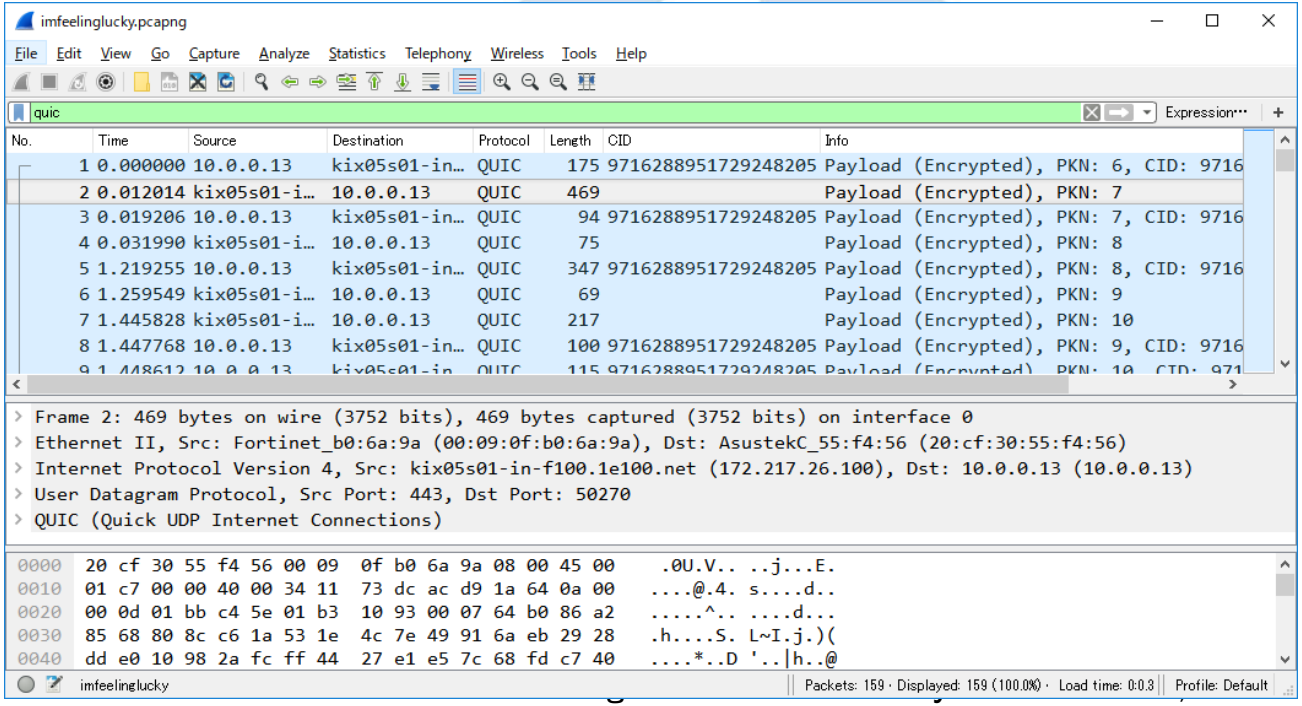


QUIC Wire Layout Specification
docs.google.com/document/d/1WJvyZflAO2pq77yOLbp9NsGjC1CHetAXV8I0fQe-B_U

Connection ID: This is an unsigned 64 bit statistically random number selected by the client that is the identifier of the connection. Because QUIC connections are designed to remain established even if the client roams, the IP 4-tuple (source IP, source port, destination IP, destination port) may be insufficient to identify the connection. For each transmission direction, when the 4-tuple is sufficient to identify the connection, the connection ID may be omitted.

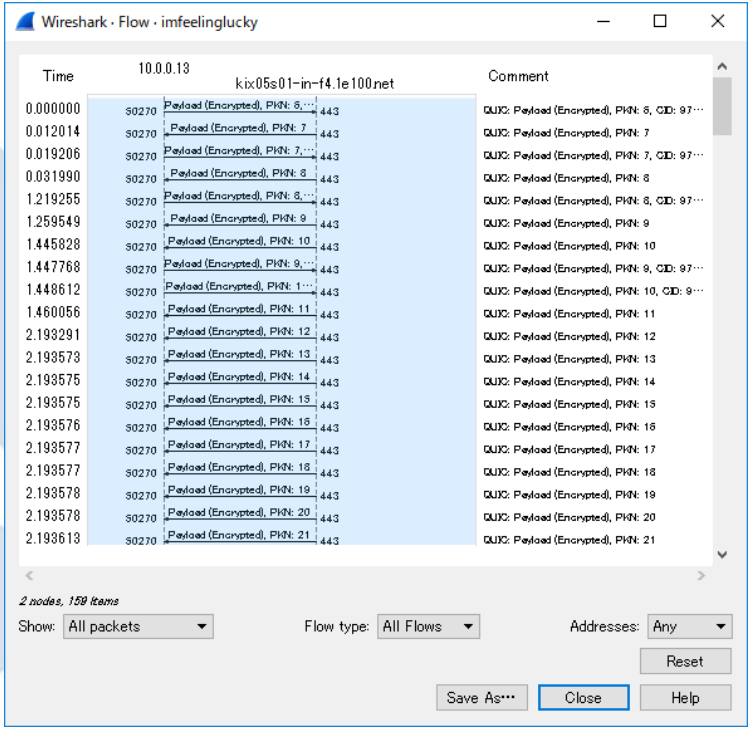
#9 QUIC dissection
Exercise 9 Add Connection ID row and filter quic

Select packet 1 and open QUIC header, and select CID (quic.cid) and right click and select “apply as column” Set quic in Display Filter



#9 QUIC dissection
Exercise 9 check 1way 2way communication in Flow

- Now Wireshark can not decrypt QUIC negotiation but we can check and find traffic of QUIC
- Create flow graph with displayed packet,
- You can find there are no 3 way handshake, There are few RTT in this connection.
- QUIC use sometimes no RTT for resumption and sometimes a few RTT for starting connection
- QUIC works in case client IP and port changed, because QUIC has NAT-Traversal mechanism



#10 Visualize using ElrasticSearch + Kibana

Wireshark dissection is the source of big pcapng analysis

- I showed this demo at last Sharkfest Europe 2016
- Wireshark is almighty decoder, packet dissection is not only for trace file analysis within Wireshark itself, but also for the source of big data analysis
- We can recode everything in network using Wireshark, and export dissection result as JSON, JSON connect Wireshark with big data analysis tools such as Elastic search and Kibana.

#10 Visualize using ElrasticSearch + Kibana

Set up Elasticsearch with Kibana environment

1. Download JDK, Curl and install, and set system environment variable
2. Download Elasticsearch and start server, check http://localhost:9200
3. Convert packet dissection data into Elasticsearch friendly JSON file
4. Entry packet dissection data (JSON) in Elasticsearch and check data
5. Modify the mapping, re-entry packet dissection data (JSON)
6. Download Kibana and start server, check http://localhost:5601
7. Access Kibana and set index
8. Search packet in full-text, visualize the packet and enjoy big data !

This time we are skipping STEP1-2, 4-7
and just enjoy Elrastic Search and Kibana

#10 Visualize using ElrasticSearch + Kibana

1. Download JDK, Curl and install, and set system environment variable

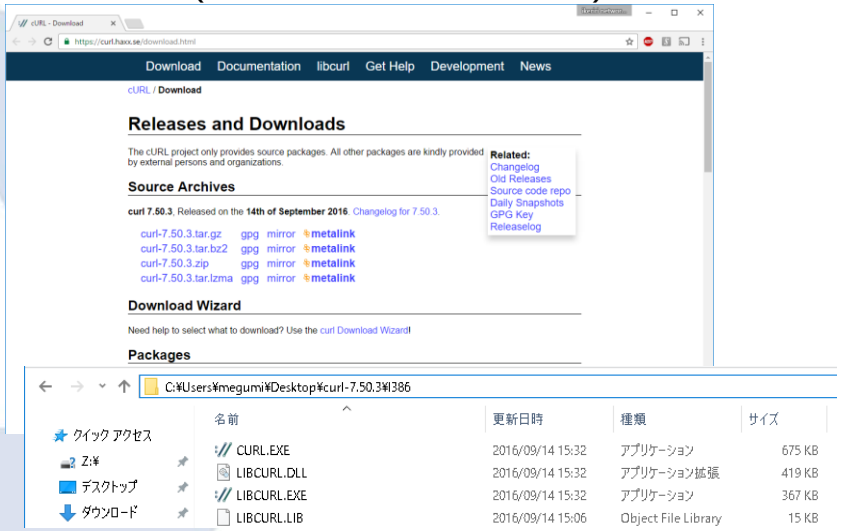
JDK(Java Developer Kit) 8 u101

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Download jdk-8u101-windows-x64.exe

Execute and start setup program

Curl (web access command)



https://curl.haxx.se/download.html

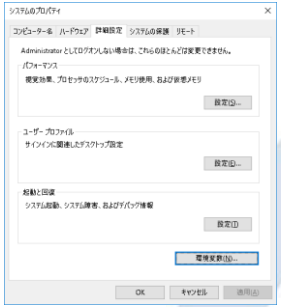
Download curl-7.50.3.cab

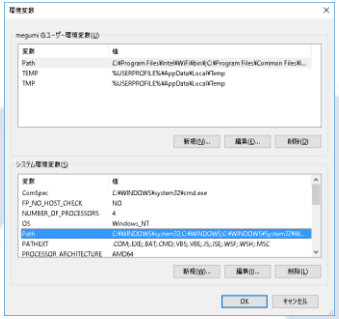
Extract cab file and copy into “Program Files”

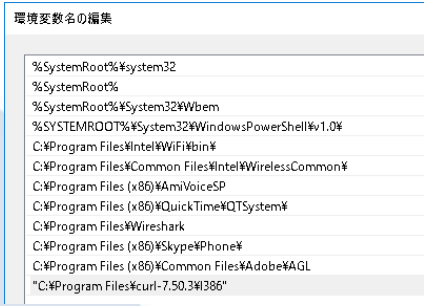
C:¥Program Files¥curl-7.50.3¥i386¥curl.exe

#10 Visualize using ElrasticSearch + Kibana

1. Download JDK, Curl and install, and set system environment variable







Control Panel > System > System setting > detail settings > environmental variable

set JAVA_HOME=C:¥Program Files¥Java¥jdk1.8.0_101

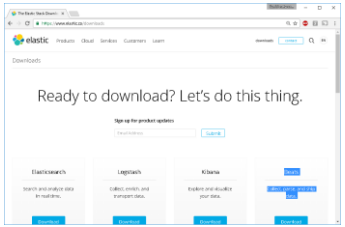
set Path=(current path);C:¥Program Files¥Java¥jdk1.8.0_101; C:¥Program Files¥curl-7.50.3¥I386

```
C:¥Users¥megumi>java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

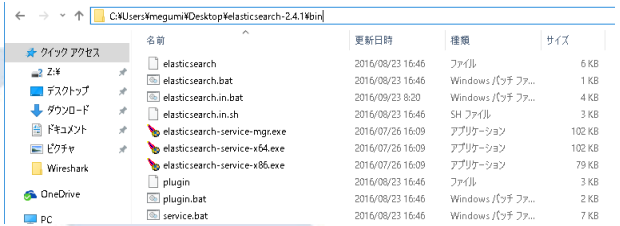
```
C:¥Users¥megumi>curl --version
curl 7.50.3 (i386-pc-win32) libcurl/7.50.3 WinSSL zlib/1.2.8
Protocols: dict file ftp ftps gopher http https imap imaps ldap pop3 pop3s rts
Features: AsynchDNS IPv6 Largefile SSPI Kerberos SPNEGO NTLM SSL libz
```

#10 Visualize using ElrasticSearch + Kibana

Download Elasticsearch and start server, check http://localhost:9200

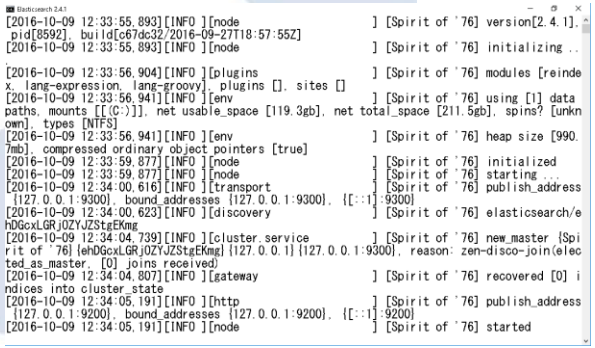


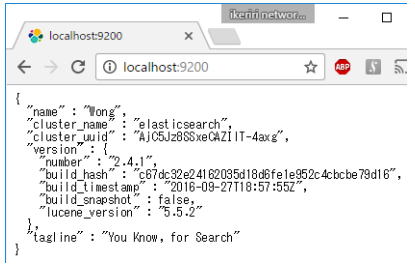
https://www.elastic.co/downloads



C:¥Users¥megumi¥Desktop¥elasticsearch-2.4.1¥bin¥elasticsearch.bat

1. Access https://www.elastic.co/downloads
2. Download elasticsearch-2.4.1.zip
3. Extract zip and open bin folder
4. Execute elasticsearch.bat
5. Check “started” in command prompt
6. Open http://localhost:9200





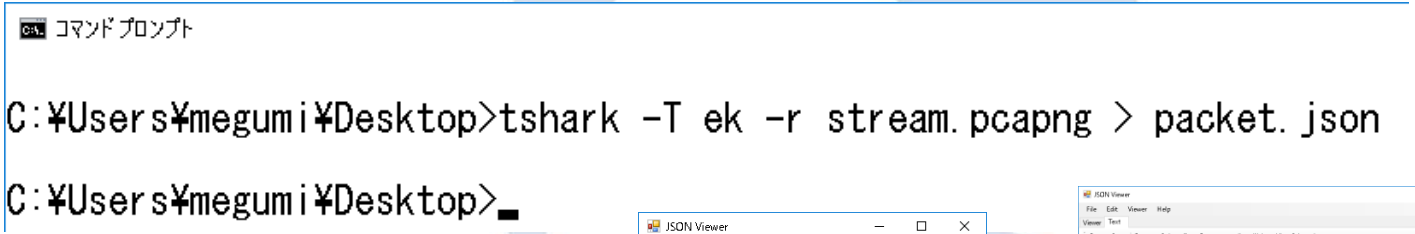
#10 Visualize using ElrasticSearch + Kibana

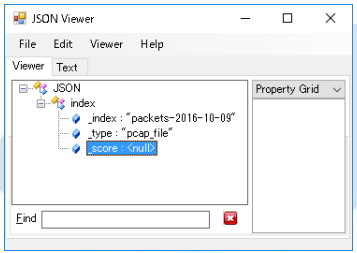
3. Convert packet dissection data into Elasticsearch friendly JSON file

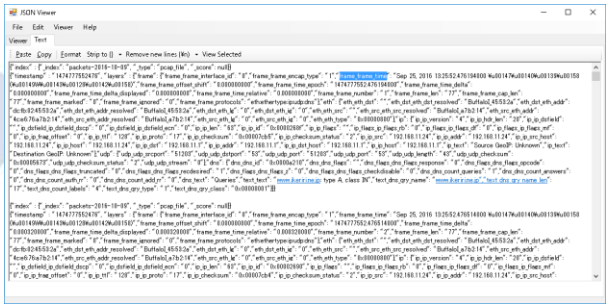
tshark -T ek –r stream.pcapng > packet.json

1: Read stream.pcapng, convert EK (Elasticsearch friendly JSON file)

2: Redirect output stream to a file named packet.json







#10 Visualize using ElrasticSearch + Kibana

4. Entry packet dissection data (JSON) in Elasticsearch and check data

curl –XPOST url @filename : use POST method to send data to server url

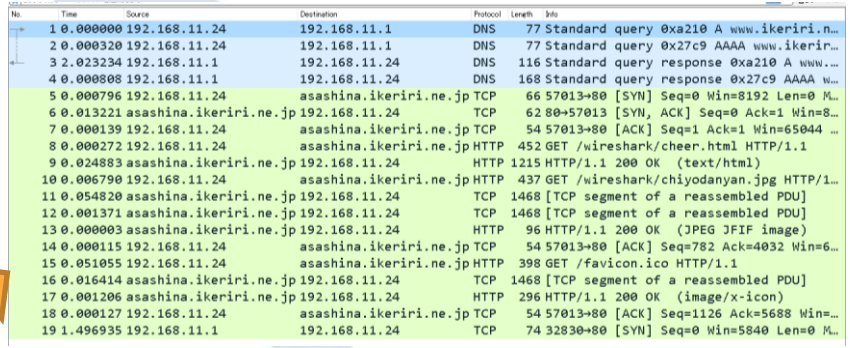
curl -XPOST http://localhost:9200/_bulk --data-binary @packet.json

C:¥Users¥megumi¥Desktop>curl -XPOST http://localhost:9200/_bulk --data-binary @packet.json
{"took":5748,"errors":false,"items":[{"create":{"_index":"packets-2016-10-09","_type":"pcap_file","_id":"AVenqQppWAnL3ZFeC-Ci","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},...
,{"create":{"_index":"packets-2016-10-09","_type":"pcap_file","_id":"AVenqQpqWAnL3ZFeC-C0","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}

#10 Visualize using ElrasticSearch + Kibana

4. Entry packet dissection data (JSON) in Elasticsearch and check data

Access http://localhost:9200/_search?pretty (_search means all index (pretty output) and check data entry




```
{
 "took" : 40,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "failed" : 0
 },
 "hits" : {
  "total" : 19,
  "max_score" : 1.0,
  "hits" : [ {
   "_index" : "packets-2016-10-09",
   "_type" : "pcap_file",
   "_id" : "AVenqQpqWAnL3ZFeC-Ck",
   "_score" : 1.0,
   "_source" : {
    "timestamp" : "1474777554499",
```

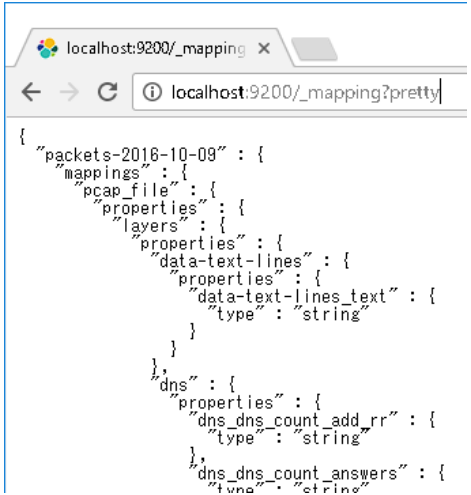
#10 Visualize using ElrasticSearch + Kibana

4. Entry packet dissection data (JSON) in Elasticsearch and check data

If you failed to entry data , use curl “curl -XDELETE http://localhost:9200/*”

```
C:¥Users¥megumi¥Desktop>curl -XDELETE http://localhost:9200/*
{"acknowledged":true}
```

Check data mappings, open browser “http://localhost:9200/_mapping?pretty”



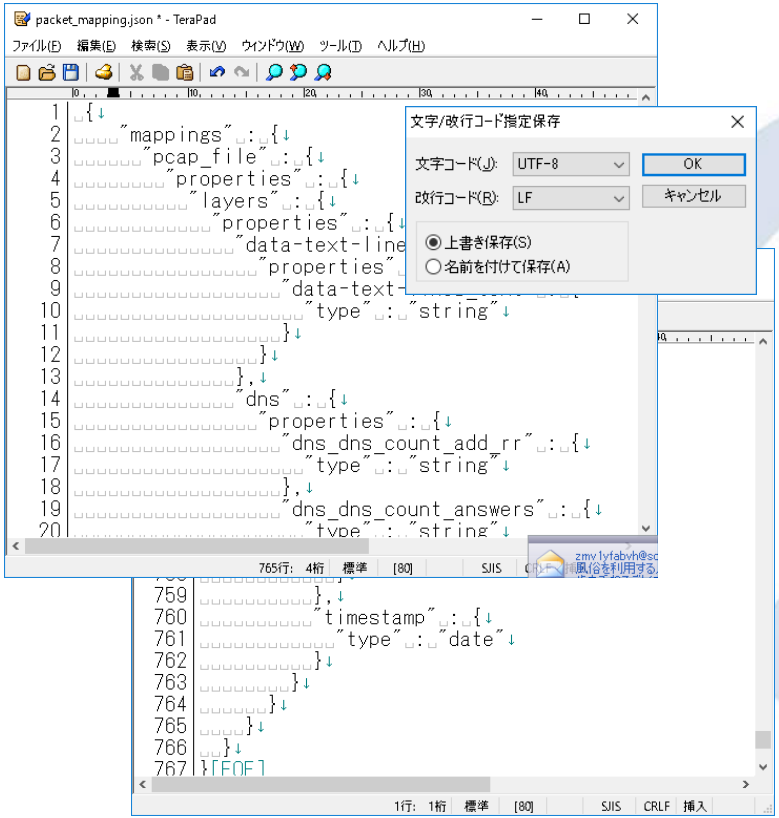

```
{
  "packets-2016-10-09" : {
    "mappings" : {
      "pcap_file" : {
        "properties" : {
          "layers" : {
            "properties" : {
              "data-text-lines" : {
                "properties" : {
                  "data-text-lines_text" : {
                    "type" : "string"
                  }
                }
              },
              "dns" : {
                "properties" : {
                  "dns_dns_count_add_rr" : {
                    "type" : "string"
                  },
```


```
                }
              }
            }
          },
          "timestamp" : {
            "type" : "string"
          }
        }
      }
    }
  }
}
```

"timestamp" : "1474777554499",

#10 Visualize using ElrasticSearch + Kibana

5. Modify the mapping, re-entry packet dissection data (JSON)



http://localhost:9200/_mapping?pretty

Save mapping as “packet_mapping.json”

Delete header **{ "packets-2016-10-09" :**

Modify mapping as

```
"timestamp" : {
    "type" : "date"
```

Check character/return code UTF-8 / LF

Delete “curl -XDELETE http://localhost:9200/*”

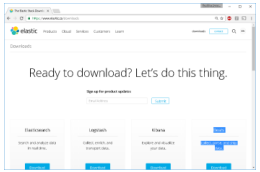
Enter mapping as **“curl -XPOST http://localhost:9200/packets-2016-10-09 --data-binary @packet_mapping.json”**

Re-enter original data

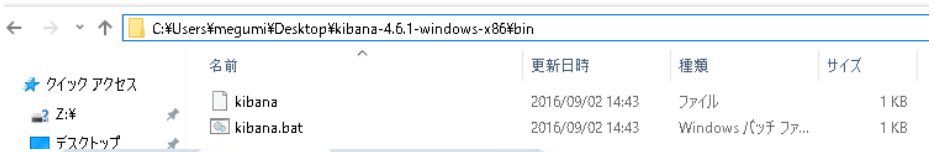
curl -XPOST http://localhost:9200/_bulk --data-binary @packet.json

#10 Visualize using ElrasticSearch ＋ Kibana

6. Download Kibana and start server, check http://localhost:5601

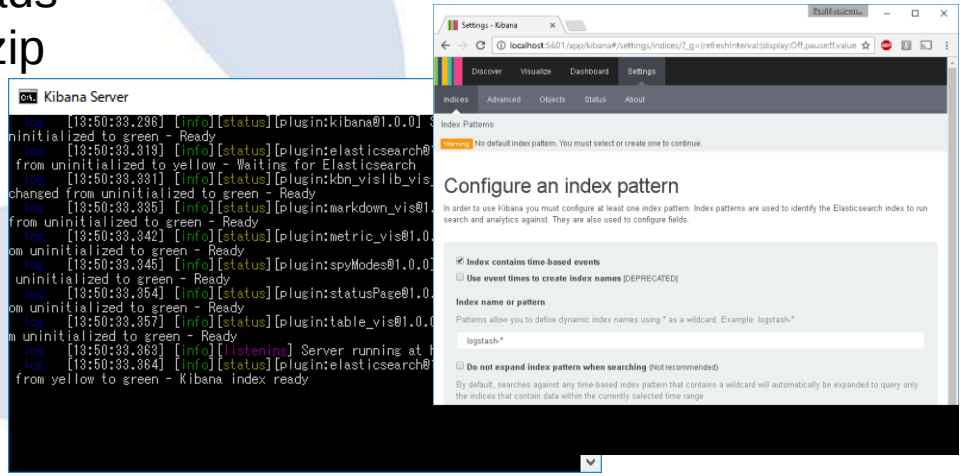


https://www.elastic.co/downloads



C:¥Users¥megumi¥Desktop¥kibana-4.6.1-windows-x86¥bin¥kibana.bat

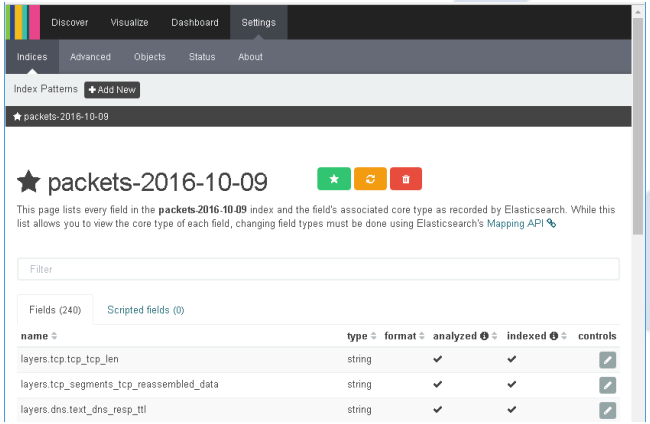
1. Access https://www.elastic.co/downloads
2. Download kibana-4.6.1-windows-x86.zip
3. Extract zip and open bin folder
4. Execute kibana.bat
5. Check “Kibana index ready”in prompt
6. Open http://localhost:5601

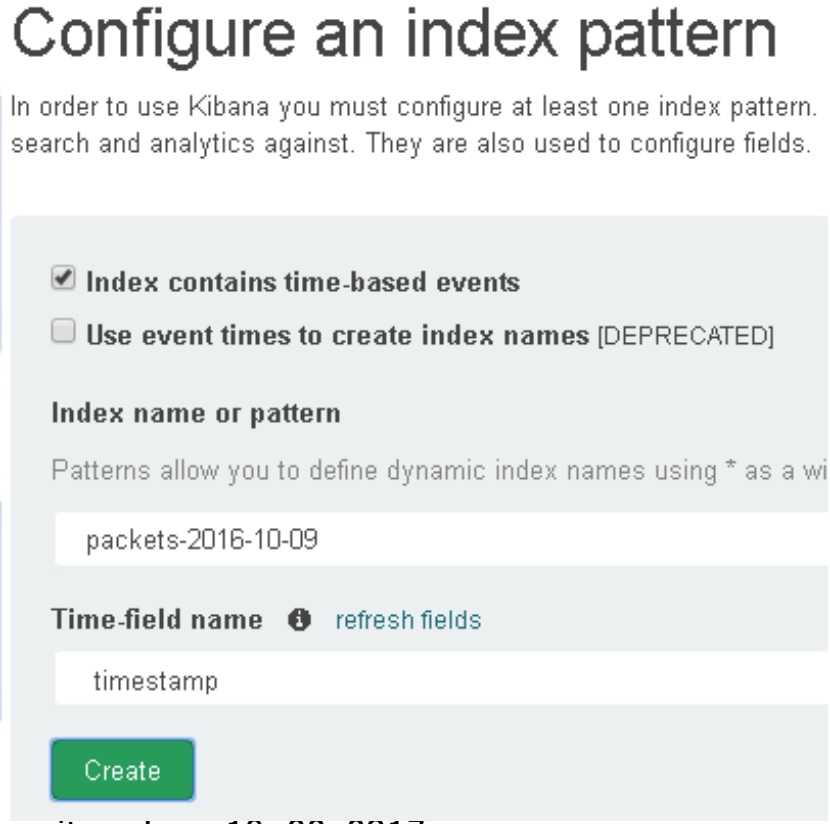


#10 Visualize using ElrasticSearch + Kibana

7. Access Kibana and set index

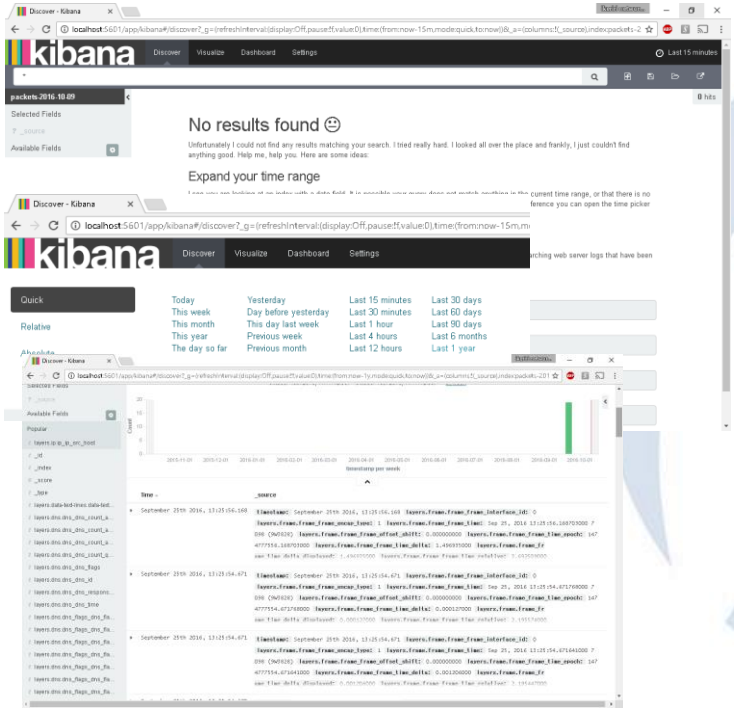
1. Access http://localhost:5601
2. Check “index contains time-based events”
3. Set Index name or pattern as
   “packets-2016-10-09” or “packets-*”
4. Set Time-field name as “timestamp”
5. Click “Create” button



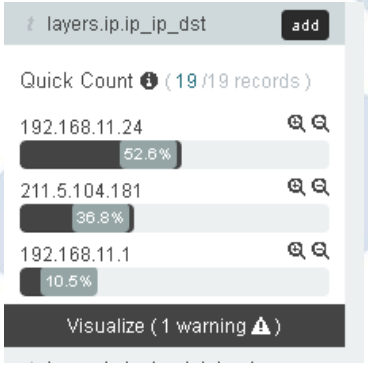


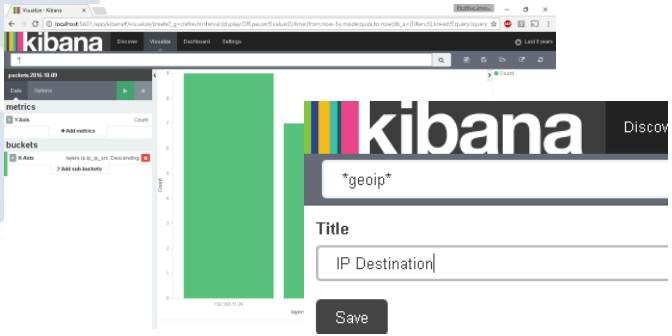
#10 Visualize using ElrasticSearch ＋ Kibana

8. Search packet in full-text, visualize the packet and enjoy big data !



1. At First, use time picker to select the time of packets (just use “Last 1 year” is a good way)
2. Check histgram and left pane
3. Select layers.ip_ip_dst in left pane, click “add” and click “Visualize”,see and save the graph as name “IP ”

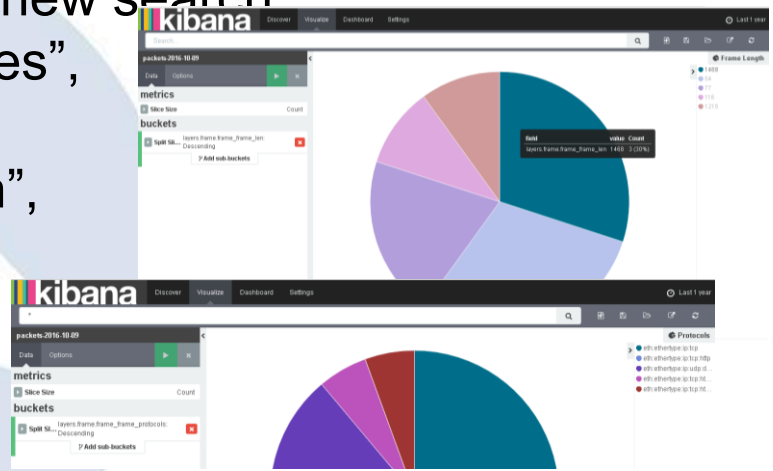




#10 Visualize using ElrasticSearch + Kibana

8. Search packet in full-text, visualize the packet and enjoy big data !

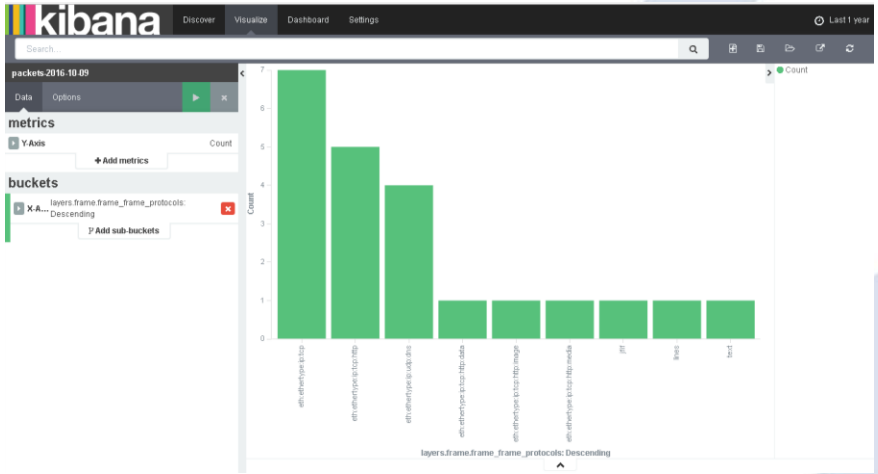
1. Select field ip.ip_ip_src, visualize and save the visualization as “IP Source”
2. Select “Visualize”, “Pie chart”, and “From a new search”, check “Select buckets type”, click “Split slices”, select “Terms” in Aggregation list box, choose field “layers.frame.frame_frame_len”, Apply changes, save the visualization as “Frame length”
3. Using “layers.frame.frame_protocol” and create pie chart, save as “Protocols”
4. Click “Dashboard” and set layout of these 4 Visualization



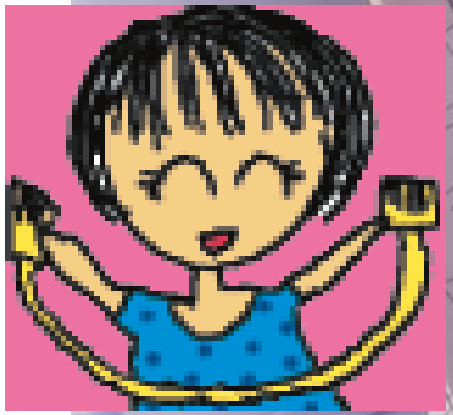
#10 Visualize using ElrasticSearch + Kibana

8. Search packet in full-text, visualize the packet and enjoy big data !

Just a few step,
We can create grate visualization
of packets, and enjoy big data !!



Finish



Use Wireshark