

SharkFest '16 Europe

Wireshark as a Spy Watermark Pen: Decrypting and Retrieving Information from Packets



16 Oct. 2016

#sf16eu

Megumi Takeshita

Packet Otaku, ikeriri network service co.,ltd.



Megumi Takeshita, ikeriri network service



- Reseller of Riverbed Technology (former CACE technologies) and Metageek, Dualcomm, Profitap and capture products in Japan
- Wrote 10+ books of Wireshark and capturing and network analysis.
- Attending all Sharkfest and translator of QT Wireshark into Japanese





Wireshark as a Spy Watermark Pen

Decrypting and Retrieving Information from Packets

- Wireshark is a good tool that provides relevant information from packets.
- In this session, demonstrate useful 8 cases of decrypting and retrieving information from packets using Wireshark, including wireless (WEP/WPA2), SSL/TLS, HTTP/SMB/TFTP, raw data, and more.



Wireshark is almighty decoder

- Decrypting and retrieving information from packet
 1. Decrypting WEP/WPA2 data
 2. Decrypting TLS/SSL data with key pair
 3. Decrypting TLS/SSL data without key pair.
 4. Retrieving Unicode Characters
 5. Retrieving object files (HTTP/TFTP/SMB)
 6. Retrieving values of field
 7. Retrieving JSON data
 8. Wireshark is the source of big data analysis !



1. Decrypting WEP/WPA2 data





1. Decrypting WEP/WPA2 data

- **WEP decryption works well only with the key**
 1. Capture packets using AirPcap or monitor mode driver
 2. Set WEP key in IEEE802.11 preference
- **WPA2 decryption needs full 4 way handshake**
 1. Capture packets using AirPcap or monitor mode driver
 2. Check complete 4 way handshake with eapol filter
 3. Set SSID and passphrase in IEEE802.11 preference



1. Decrypting WEP/WPA2 data

1. Capture packets using AirPcap or monitor mode driver
wep.pcapng(<http://www.ikeriri.ne.jp/temp/>)

No.	Time	Source	Destination	Protocol	Length	Info
1	13:13:29.989012	PlanexCo_e3:c2:...	Broadcast	802.11	108	Beacon frame, SN=25...
2	13:13:30.030908	IntelCor_1b:a1:...	Modacom_3d:9c:b8	802.11	120	Data, SN=256, FN=0,...
3	13:13:30.030932		IntelCor_1b:a1:...	802.11	34	Acknowledgement, FL...
4	13:13:30.032854	OkiElect_f6:f4:...	Broadcast	802.11	109	Beacon frame, SN=61...
5	13:13:30.041460	41:ba:00:e3:c2:...	54:5e:dc:96:91:...	802.11	180	Unrecognized (Reser...
6	13:13:30.041483		PlanexCo_e3:c2:...	802.11	34	Acknowledgement, FL...
7	13:13:30.041856	00:49:5f:86:aa:...	a0:68:b4:1b:a1:...	802.11	180	Data, SN=2623, FN=1...
8	13:13:30.041883		PlanexCo_e3:c2:...	802.11	34	Acknowledgement, FL...

> Frame 1: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface 0
> Radiotap Header v0, Length 20
> 802.11 radio information
> IEEE 802.11 Beacon frame, Flags:C
> IEEE 802.11 wireless LAN management frame

```
0000 00 00 14 00 ee 18 00 00 10 02 7b 09 a0 00 ce 9c ..... ..{.....
0010 64 00 00 32 80 00 00 00 ff ff ff ff ff ff 00 90 d..2.....
0020 cc e3 c2 79 00 90 cc e3 c2 79 b0 9c 9b 21 1b 8c ...y...y...!
0030 01 00 00 00 64 00 11 04 00 09 63 6c 65 61 72 74 ....d...c...t
0040 65 78 74 01 08 82 84 8b 96 0c 12 18 24 03 01 04 ext.....
0050 05 04 00 01 00 00 2a 01 04 32 04 30 48 60 6c dd
```

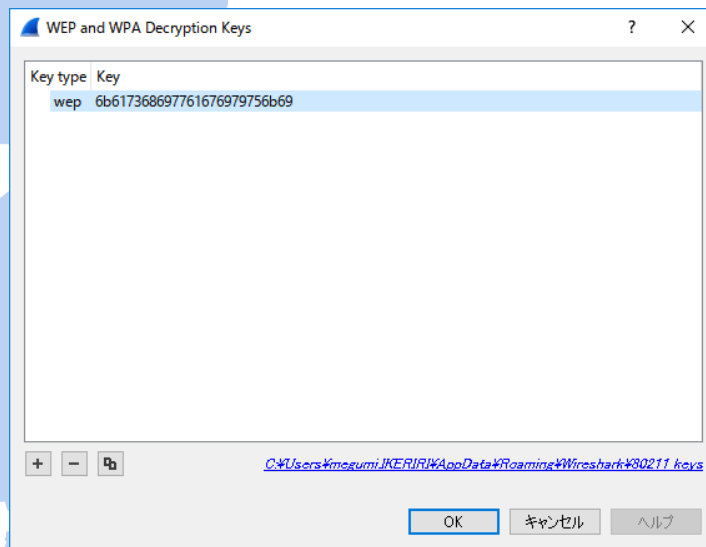
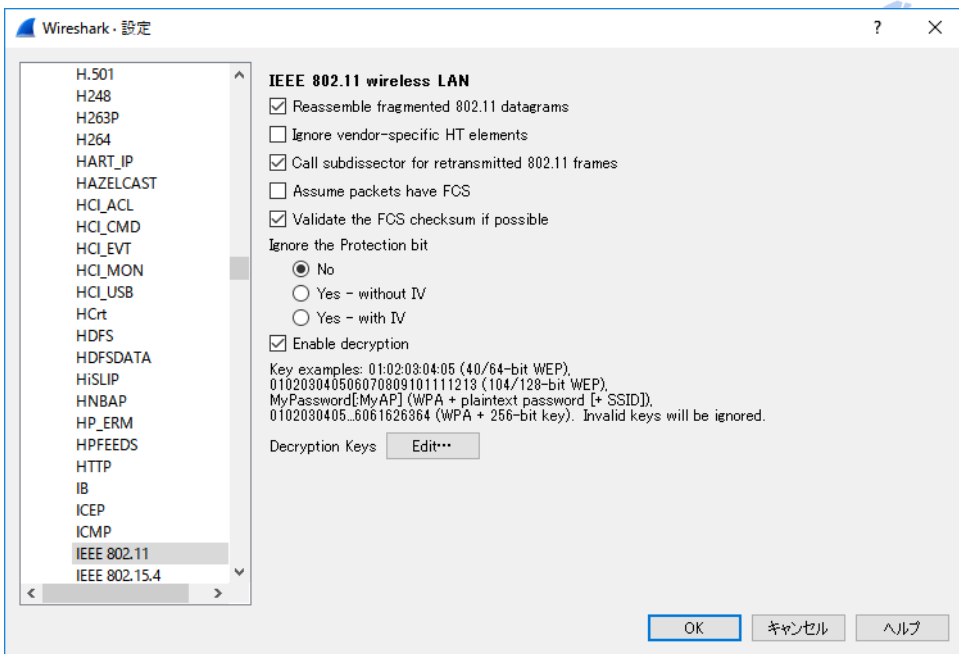
wep | パケット数: wep.pcapng



1. Decrypting WEP/WPA2 data

2. Set WEP key in IEEE802.11 preference

WEP key : kashiwagi yuki (6b617368697761676979756b69)





1. Decrypting WEP/WPA2 data

Check #458 in trace file wep.pcapng

Packet #458 details:

No.	Time	Source	Destination	Protocol	Length	Info
458	13:13:35.534747	Modacom_3d:9c:b8	IntelCor_1b:a1...	802.11	524	Data, SN=2620, FN=0...
459	13:13:35.534767		PlanexCo_e3:c2...	802.11	34	Acknowledgement, Fl...
460	13:13:35.535439	Modacom_ed:93:b8	IntelCor_1b:a1...	802.11	241	Data, SN=275, FN=0,...
461	13:13:35.535459		IntelCor_1b:a1...	802.11	34	Acknowledgement, Fl...

Frame 458: 524 bytes on wire (4192 bits), 524 bytes captured (4192 bits) on interface 0

- Radiotap Header v0, Length 20
- 802.11 radio information
- IEEE 802.11 Data, Flags: .p...F.C
- Data (468 bytes)

Packet bytes:

```
0010 56 00 00 34 08 42 2c 00 a0 88 b4 1b a1 f0 00 9e V..4.B, .....
0020 cc e3 c2 79 00 1d 93 3d 9c b8 c0 a3 aa aa 03 00 ...y...= .....
0030 70 f0 91 87 eb e4 b2 10 b7 18 5f 91 b8 91 81 a5 p.....
0040 76 04 e5 38 63 6f df 12 71 83 a9 e9 ef 9c d7 e9 v..Bco. q.....
0050 2a 25 65 fe 59 71 71 b5 1c e9 49 af d2 45 d4 00 *%e.Yqq. .I..E..
0060 f7 c6 64 49 47 95 95 5a e3 37 f1 fa bb 9c 1a 2b ..dIG..Z .7.....+
0070 8c 9f 45 48 ea dc 60 54 9f 5c 06 64 56 de 4a 7d ..EH..T .\d.V.J}
0080 17 cc 3e 9b 2d d1 c1 ea 38 d2 8f e5 45 98 50 78 >.-... 8...E.Px
```



Packet #458 details:

No.	Time	Source	Destination	Protocol	Length	Info
458	13:13:35.534747	192.168.100.254	192.168.100.104	HTTP	524	NOTIFY /upnp/eventi...
459	13:13:35.534767		PlanexCo_e3:c2...	802.11	34	Acknowledgement, Fl...
460	13:13:35.535439	192.168.100.104	192.168.100.254	HTTP	241	HTTP/1.1 200 OK
461	13:13:35.535459		IntelCor_1b:a1...	802.11	34	Acknowledgement, Fl...

Frame 458 details:

- IEEE 802.11 Data, Flags: .p....TC
- Logical-Link Control
- Internet Protocol Version 4, Src: 192.168.100.104 (192.168.100.104), Dst: 192.168.100.254 (192.168.100.254)
- Transmission Control Protocol, Src Port: 2869, Dst Port: 3517, Seq: 1, Ack: 623, Len: 125
- Hypertext Transfer Protocol
 - HTTP/1.1 200 OK\r\n
 - Server: Microsoft-HTTPAPI/2.0\r\n
 - Date: Thu, 26 Jun 2014 04:13:18 GMT\r\n
 - Connection: close\r\n
 - Content-Length: 0\r\n
 - \r\n

Packet bytes:

```
0000 aa aa 03 00 00 08 00 45 00 00 b1 3c 90 40 00 ..... E...<.@.
0010 80 06 72 ff c0 a8 64 68 c0 a8 64 fe 0b 35 0d bd ...r...dh ...d..5..
0020 3a 2e 85 24 db e8 eb 4d 80 19 00 42 ec 46 00 00 :...$.M...B.F..
0030 01 01 08 0a 01 67 61 d1 00 89 9f 7f 48 54 54 50 :...ga....HTTP
0040 2f 31 2e 31 20 32 30 30 20 4f 4b 0d 0a 53 65 72 /1.1 200 OK..Ser
0050 76 65 72 3a 20 4d 69 63 72 6f 73 6f 66 74 2d 48 ver: Mic rosoft-H
0060 54 54 50 41 50 49 2f 32 2e 30 0d 0a 44 61 74 65 TTPAPI/2 .0..Date
```



1. Decrypting WEP/WPA2 data

1. Capture packets using AirPcap or monitor mode driver
wpa2.pcapng(<http://www.ikeriri.ne.jp/temp/>)

No.	Time	Source	Destination	Protocol	Length	Info
1	13:29:40.585335	PlanexCo_e3:c2:...	Broadcast	802.11	130	Beacon frame, SN=1597, FN=0, Flags=...
2	13:29:40.664448	IntelCor_1b:a1:...	Broadcast	802.11	103	Probe Request, SN=969, FN=0, Flags=...
3	13:29:40.794031	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	802.11	124	Probe Response, SN=1603, FN=0, Flags=...
4	13:29:40.899226	IntelCor_1b:a1:...	PlanexCo_e3:c2:...	802.11	54	Authentication, SN=3565, FN=0, Flags=...
5	13:29:40.899256		IntelCor_1b:a1:...	802.11	34	Acknowledgement, Flags=.....C
6	13:29:40.900019	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	802.11	54	Authentication, SN=1607, FN=0, Flags=...
7	13:29:40.900416		PlanexCo_e3:c2:...	802.11	34	Acknowledgement, Flags=.....C
8	13:29:40.908150	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	802.11	70	Association Response, SN=1611, FN=0, ...
9	13:29:40.908670		PlanexCo_e3:c2:...	802.11	34	Acknowledgement, Flags=.....C
10	13:29:40.909315	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	802.11	70	Association Response, SN=1611, FN=0, ...
11	13:29:40.910280		PlanexCo_e3:c2:...	802.11	34	Acknowledgement, Flags=.....C
12	13:29:40.914008	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	EAPOL	177	Key (Message 1 of 4)

The packet details pane for the selected frame shows:

- > Frame 1: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits)
- > Radiotap Header v0, Length 20
- > 802.11 radio information
- > IEEE 802.11 Beacon frame, Flags:C
- > IEEE 802.11 wireless LAN management frame

The packet bytes pane shows the raw hex and ASCII data for the selected frame:

```
0000 00 00 14 00 ee 18 00 00 10 02 7b 09 a0 00 d0 9c ..... {f.....
0010 64 00 00 34 80 00 00 00 ff ff ff ff ff 00 90 d..4.....
0020 cc e3 c2 79 00 90 cc e3 c2 79 d0 63 ec 71 65 25 ...y.... .y.c.qe%
0030 00 00 00 00 64 00 11 04 00 09 63 6c 65 61 72 74 ...d.... cleart
0040 65 78 74 01 08 82 84 8b 96 0c 12 18 24 03 01 04 ext.....$....
0050 05 04 00 01 00 00 2a 01 04 32 04 30 48 60 6c 30 .....*.20H'10
0060 14 01 00 00 0f ac 04 01 00 00 0f ac 04 01 00 00 .....
0070 0f ac 02 00 00 dd 07 00 e0 4c 01 02 03 00 84 2d ..... .L.....
0080 3d 93 =.
```

wpa2.pcapng



1. Decrypting WEP/WPA2 data

2. Check complete 4 way handshake with eapol filter

Note: Trace file must contain complete set of 4 way handshake

The screenshot shows a Wireshark capture of a 4-way EAPOL handshake. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
12	13:29:40.914008	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	EAPOL	177	Key (Message 1 of 4)
13	13:29:40.914009	IntelCor_1b:a1:...	PlanexCo_e3:c2:...	EAPOL	179	Key (Message 2 of 4)
14	13:29:40.921648	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	EAPOL	211	Key (Message 3 of 4)
15	13:29:40.922259	IntelCor_1b:a1:...	PlanexCo_e3:c2:...	EAPOL	155	Key (Message 4 of 4)

The details pane for frame 12 shows the following structure:

- > Frame 12: 177 bytes on wire (1416 bits), 177 bytes captured (1416 bits)
- > Radiotap Header v0, Length 20
- > 802.11 radio information
- > IEEE 802.11 Data, Flags:R.F.C
- > Logical-Link Control
- > 802.1X Authentication
 - Version: 802.1X-2001 (1)
 - Type: Key (3)
 - Length: 117
 - Key Descriptor Type: EAPOL RSN Key (2)
 - Key Information: 0x008a
 - Key Length: 16
 - Replay Counter: 0
 - WPA Key Nonce: a2ad060d95ffa0c473f8d89213da7e7f2ad0e54b990b0a92...

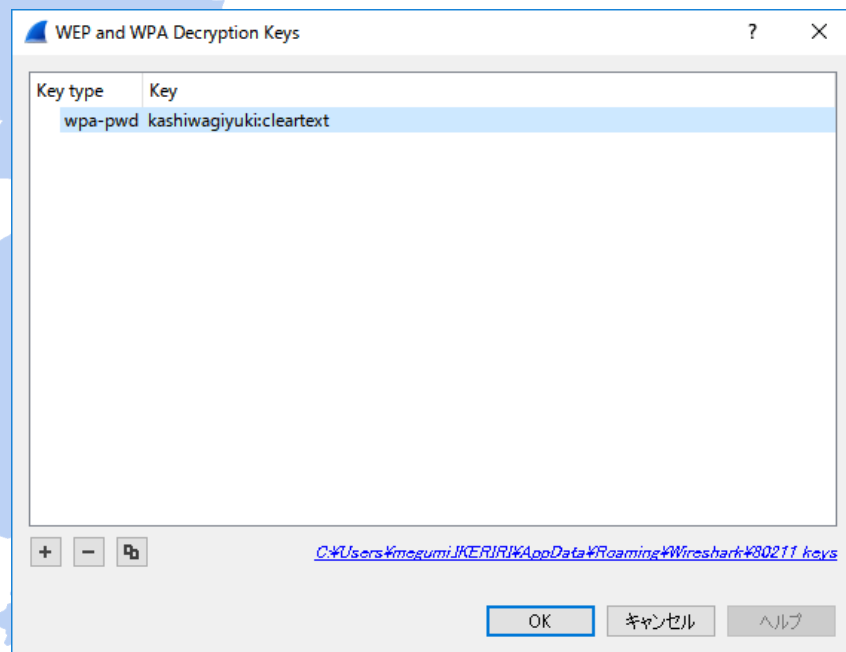
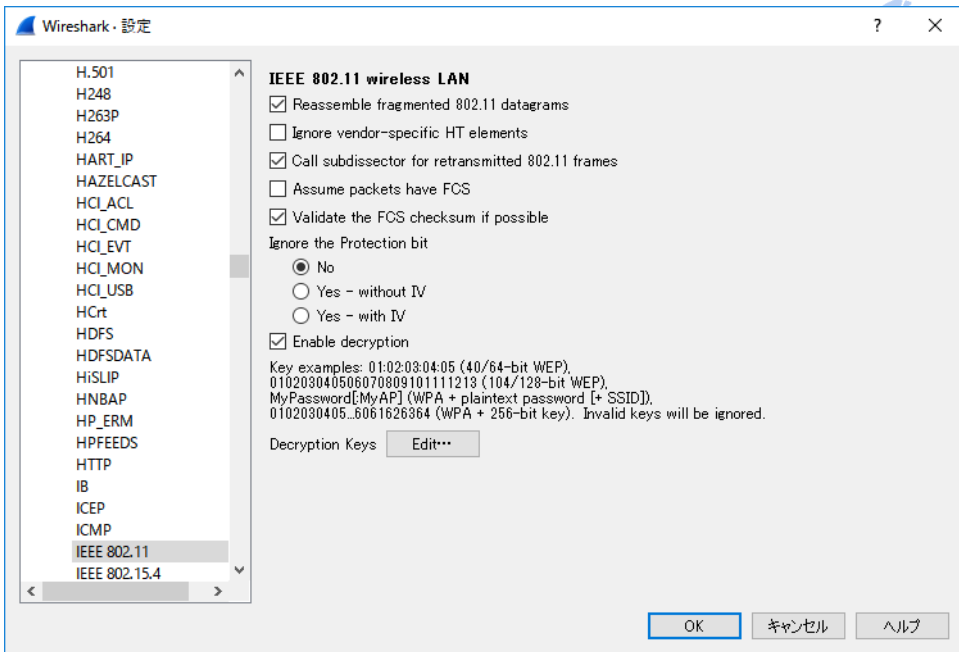
The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII column shows the start of the WPA Key Nonce: ".usKD&k2.q TA. 4..2[... @".



1. Decrypting WEP/WPA2 data

3. Set SSID and passphrase in IEEE802.11 preference

Passphrase : kashiwagi yuki SSID : cleartext





1. Decrypting WEP/WPA2 data

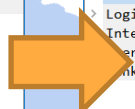
Check #16 in trace file wpa2.pcapng

The left screenshot shows the Wireshark interface for the file wpa2.pcapng. The packet list pane shows several packets, with packet #16 selected and highlighted in blue. The packet bytes pane shows the raw data of the selected packet, with a blue highlight over the CCMP data field.

No.	Time	Source	Destination	Protocol	Length	Info
14	13:29:40.921648	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	EAPOL	211	Key (Message 3 of 4)
15	13:29:40.922259	IntelCor_1b:a1:...	PlanexCo_e3:c2:...	EAPOL	155	Key (Message 4 of 4)
16	13:30:16.230648	IntelCor_1b:a1:...	IPv6mcast_01:0:...	802.11	144	Data, SN=243, FN=0, Flags=.p....TC
17	13:30:16.230914	IntelCor_1b:a1:...	IPv6mcast_01:0:...	802.11	144	Data, SN=243, FN=0, Flags=.p..R..TC
18	13:30:16.231742	IntelCor_1b:a1:...	IPv6mcast_01:0:...	802.11	144	Data, SN=243, FN=0, Flags=.p..R..TC

Frame 16: 144 bytes on wire (1152 bits), 144 bytes captured (1152 bits)
> Radiotap Header v0, Length 20
> 802.11 radio information
> IEEE 802.11 Data, Flags: .p....TC
> Data (88 bytes)

```
0000 00 00 14 00 ee 18 00 00 10 24 7b 09 c0 00 e9 9c .....$.{.....
0010 3a 00 00 4d 08 41 30 00 00 90 cc e3 c2 79 a0 88 ...M.A0.....y...
0020 b4 1b a1 f0 33 33 00 01 00 03 30 0f ec 02 00 20 ...33...0.....
0030 00 00 00 00 c1 3c 4e 50 81 88 d7 9a 4b 28 1e 4c ...<NP...K(L...
0040 0d b3 6b dd 2c 29 de 46 43 98 84 19 dc 00 4b 0d ...k.,).F C...K...
0050 79 eb 41 b8 c2 53 ff c9 02 7d d7 f7 26 9a b7 7e y.A..S...}.&...
0060 ee 39 c7 76 80 6a bf b3 7b 6c d1 38 bf 1c 8d f0 ..9.v.j..{1.8....
0070 10 62 c8 a2 bd 48 1b b1 cb 43 cf d6 65 0d d8 f3 ..b..H...C.e....
0080 be b1 13 02 a8 07 82 fa 79 08 21 6d 71 37 51 41 .....y.!mq7QA
```



The right screenshot shows the Wireshark interface for the file wpa2.pcapng. The packet list pane shows several packets, with packet #16 selected and highlighted in blue. The packet bytes pane shows the raw data of the selected packet, with a blue highlight over the CCMP data field. The packet bytes pane also shows the decrypted CCMP data.

No.	Time	Source	Destination	Protocol	Length	Info
14	13:29:40.921648	PlanexCo_e3:c2:...	IntelCor_1b:a1:...	EAPOL	211	Key (Message 3 of 4)
15	13:29:40.922259	IntelCor_1b:a1:...	PlanexCo_e3:c2:...	EAPOL	155	Key (Message 4 of 4)
16	13:30:16.230648	fe80::f017:b974:...	ff02::1:3	LLMNR	144	Standard query 0x064e A isatap
17	13:30:16.230914	fe80::f017:b974:...	ff02::1:3	LLMNR	144	Standard query 0x064e A isatap
18	13:30:16.231742	fe80::f017:b974:...	ff02::1:3	LLMNR	144	Standard query 0x064e A isatap

Frame 16: 144 bytes on wire (1152 bits), 144 bytes captured (1152 bits)
> Radiotap Header v0, Length 20
> 802.11 radio information
> IEEE 802.11 Data, Flags: .p....TC
> Logical-Link Control
> Internet Protocol Version 6, Src: fe80::f017:b974:7130:eb31 (fe80::f017:b974:7130:eb31), Dst: ff02::1:3 (ff02::1:3)
> User Datagram Protocol, Src Port: 59500, Dst Port: 5355
> Multicast Local Multicast Name Resolution (query)

```
0000 aa aa 03 00 00 00 86 dd 60 00 00 00 00 20 11 01 .....`.....
0010 fe 80 00 00 00 00 00 f0 17 b9 74 71 30 eb 31 .....tq0.1
0020 ff 02 00 00 00 00 00 00 00 00 00 01 00 01 00 03 .....
0030 e8 6c 14 eb 00 20 9a 62 06 4e 00 00 00 01 00 00 ..l...b.N.....
0040 00 00 00 00 06 69 73 61 74 61 70 00 00 01 00 01 .....isa tap....
```

Frame (144 bytes) Decrypted CCMP data (80 bytes)
IEEE 802.11 wireless LAN (wlan), 32 バイト



2. Decrypting TLS/SSL data with key pair





2. Decrypting TLS/SSL data with key pair

1. PEM format file (certification with public and private key) needed for decryption
Collect and convert information from server
2. Capture packets including SSL
Check SSL/TLS handshake in a trace file
3. Set server ip, tcp port, protocol and key(PEM file) in SSL preference



2. Decrypting TLS/SSL data with key pair

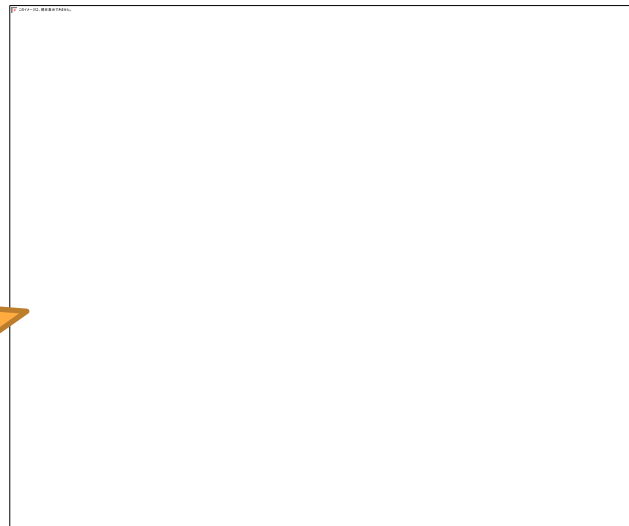
1. PEM format file (certification with public and private key) needed for decryption

Collect and convert information from server



IP address : 192.168.100.200 TCP port : 443
Apache config : /etc/apache2/httpd.conf
SSL config : /etc/apache2/sites-available/httpd-ssl.conf
cert file : /etc/apache2/ssl/cert-file.crt

PEM (Privacy-enhanced Electronic Mail)
contains server private and public key





2. Decrypting TLS/SSL data with key pair

- 2. Capture packets including SSL (<https://192.168.100.200>) ssl.pcapng
Check SSL/TLS handshake in a trace file

No.	Time	Source	Destination	Protocol	Length	Info
1	20:00:01.438323	192.168.100.122	192.168.100.200	TCP	66	10167→443 [SYN] Seq=0 Win=8192 Len=0
2	20:00:01.438744	192.168.100.200	192.168.100.122	TCP	66	443→10167 [SYN, ACK] Seq=0 Ack=1 Win=
3	20:00:01.438806	192.168.100.122	192.168.100.200	TCP	54	10167→443 [ACK] Seq=1 Ack=1 Win=65536
4	20:00:01.438935	192.168.100.122	192.168.100.200	TCP	66	10168→443 [SYN] Seq=0 Win=8192 Len=0
5	20:00:01.439215	192.168.100.200	192.168.100.122	TCP	66	443→10168 [SYN, ACK] Seq=0 Ack=1 Win=
6	20:00:01.439224	192.168.100.122	192.168.100.200	TLSv1	571	Client Hello
7	20:00:01.439260	192.168.100.122	192.168.100.200	TCP	54	10168→443 [ACK] Seq=1 Ack=1 Win=65536
8	20:00:01.439474	192.168.100.122	192.168.100.200	TLSv1	571	Client Hello
9	20:00:01.439681	192.168.100.200	192.168.100.122	TCP	60	443→10167 [ACK] Seq=1 Ack=518 Win=691
10	20:00:01.439787	192.168.100.200	192.168.100.122	TCP	60	443→10168 [ACK] Seq=1 Ack=518 Win=69
11	20:00:01.444222	192.168.100.200	192.168.100.122	TLSv1	199	Server Hello, Change Cipher Spec, Enc...
12	20:00:01.445068	192.168.100.122	192.168.100.200	TLSv1	113	Change Cipher Spec, Encrypted Handsh...

> Frame 12: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: Inventec_2f:9c:c6 (00:8c:fa:2f:9c:c6), Dst: AvalueTe_02:0b:79 (00:04:5f:02:0b:79)
> Internet Protocol Version 4, Src: 192.168.100.122 (192.168.100.122), Dst: 192.168.100.200 (192.168.100.200)
> Transmission Control Protocol, Src Port: 10167, Dst Port: 443, Seq: 0, Len: 0

```
0000 00 04 5f 02 0b 79 00 8c fa 2f 9c c6 08 00 45 00  . . . y . . / . . . E . . .
0010 00 3a 3a 0b 40 00 80 06 00 00 c0 a8 64 7a c0 a8  . . : @ . . . . d z . .
0020 64 c8 27 b7 01 bb 14 74 20 95 00 00 00 80 02  . d . . . . . t . . . .
0030 20 00 4a ba 00 00 02 04 05 b4 01 03 03 08 01 01  . J . . . . . . . . . .
0040 04 02  . . . . . . . . . . . . . . . . . . . . . . . .
```

ssl.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
17	20:00:01.450202	192.168.100.200	192.168.100.122	TLSv1	91	Encrypted Alert
24	20:00:01.451401	192.168.100.200	192.168.100.122	TLSv1	91	Encrypted Alert
28	20:00:01.451705	192.168.100.122	192.168.100.200	TLSv1	571	Client Hello
30	20:00:01.455306	192.168.100.200	192.168.100.122	TLSv1	199	Server Hello, Change Cipher Spec, Enc...
31	20:00:01.455763	192.168.100.122	192.168.100.200	TLSv1	113	Change Cipher Spec, Encrypted Handsha...
32	20:00:01.455949	192.168.100.122	192.168.100.200	TLSv1	512	Application Data, Application Data
34	20:00:01.460474	192.168.100.200	192.168.100.122	TLSv1	666	Application Data, Application Data, A...
35	20:00:01.482279	192.168.100.122	192.168.100.200	TLSv1	416	Application Data, Application Data, A...
36	20:00:01.485205	192.168.100.200	192.168.100.122	TLSv1	682	Application Data, Application Data, A...
41	20:00:03.120765	192.168.100.122	192.168.100.200	TLSv1	571	Client Hello
43	20:00:03.122240	192.168.100.122	192.168.100.200	TLSv1	512	Application Data, Application Data
44	20:00:03.124332	192.168.100.200	192.168.100.122	TLSv1	199	Server Hello, Change Cipher Spec, Enc...

> Frame 32: 512 bytes on wire (4096 bits), 512 bytes captured (4096 bits) on interface 0
> Ethernet II, Src: Inventec_2f:9c:c6 (00:8c:fa:2f:9c:c6), Dst: AvalueTe_02:0b:79 (00:04:5f:02:0b:79)
> Internet Protocol Version 4, Src: 192.168.100.122 (192.168.100.122), Dst: 192.168.100.200 (192.168.100.200)
> Transmission Control Protocol, Src Port: 10169, Dst Port: 443, Seq: 577, Ack: 146, Len: 458
> Secure Sockets Layer

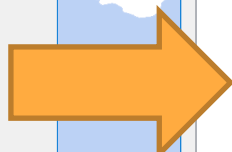
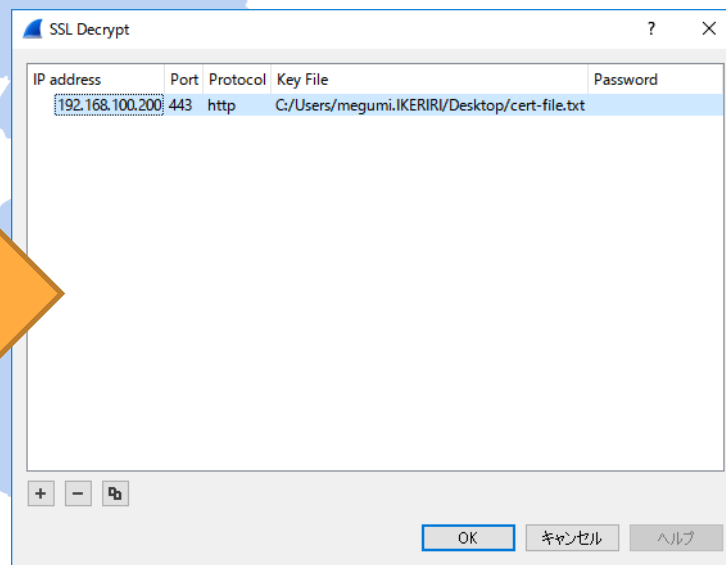
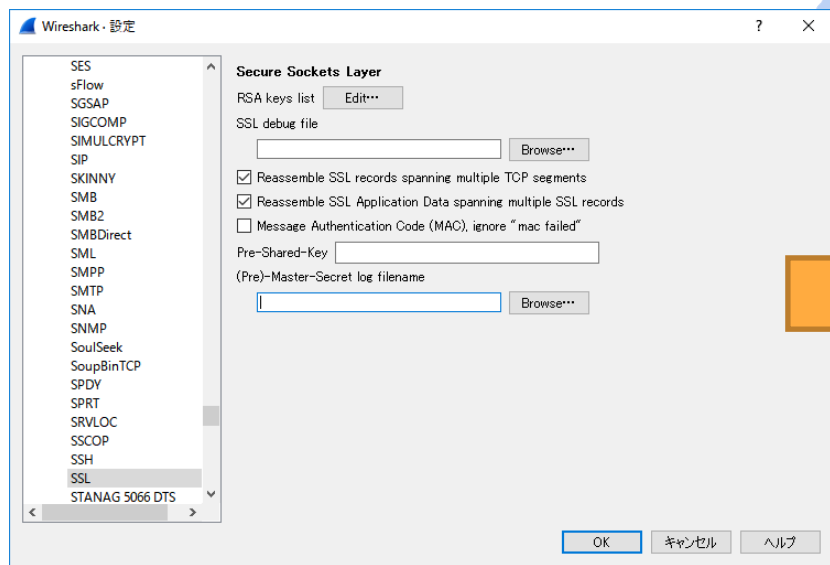
```
0000 00 04 5f 02 0b 79 00 8c fa 2f 9c c6 08 00 45 00  . . . y . . / . . . E . . .
0010 01 f2 3a 1d 40 00 80 06 00 00 c0 a8 64 7a c0 a8  . . : @ . . . . d z . .
0020 64 c8 27 b9 01 bb 83 27 d6 76 2d 04 51 df 50 18  . d . . . . . v . . . . Q . P .
0030 01 00 4c 78 00 00 17 03 01 00 20 7e f9 25 6e 3a  . L . x . . . . . . . . . .
0040 4d 9a eb df 35 d8 cb 40 f4 30 c9 5a d4 59 48 65  . M . . . . . @ . . Z . Y % .
0050 ee 84 b8 42 34 dc a7 03 e2 ff 17 17 03 01 01 a0  . . . B4 . . . . . . . . . .
0060 d3 e0 1d 2f 74 51 aa 7e 98 7f 55 aa d3 ec eb ea  . . . / t Q . . . . U . . . .
0070 09 2c b1 51 fa 7d 9f 75 1c 18 3c 28 57 28 f0 36  . . . Q . . u . . < ( W ( . 6
0080 23 44 ea 63 7b ee 5f e4 29 a0 5e 9d b2 96 75 d7  . # D . c ( . . ) . ^ . . . . u .
```



2. Decrypting TLS/SSL data with key pair

3. Set server ip, tcp port, protocol and key(PEM file) in SSL preference

192.168.100.200,443,http,cert-file.txt





2. Decrypting TLS/SSL data with key pair

Check #90,91 in trace file ssl.pcapng

ssl.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
88	20:00:20...	192.168.1...	192.168.1...	TLSv1	320	New Session Ticket, Change Cipher Spec, ...
89	20:00:20...	192.168.1...	192.168.1...	TCP	54	10189→443 [ACK] Seq=355 Ack=780 Win=2611...
90	20:00:20...	192.168.1...	192.168.1...	TLSv1	400	Application Data, Application Data
91	20:00:20...	192.168.1...	192.168.1...	TLSv1	666	Application Data, Application Data, Appl...

> Frame 90: 400 bytes on wire (3200 bits), 400 bytes captured (3200 bits) on interface 0
> Ethernet II, Src: Inventec_2f:9c:c6 (00:8c:fa:2f:9c:c6), Dst: AvalueTe_02:0b:79 (00:04:5f:02:0b:79)
> Internet Protocol Version 4, Src: 192.168.100.122 (192.168.100.122), Dst: 192.168.100.200 (192.168.100.200)
> Transmission Control Protocol, Src Port: 10189, Dst Port: 443, Seq: 355, Ack: 780, Len: 346
> Secure Sockets Layer

```
0030 03 fc 4c 08 00 00 17 03 01 00 20 b6 09 d1 90 2f ...L... ..  
0040 e8 f4 1a 9f be 51 5c 70 c0 e8 11 47 e5 21 a4 6f ...Qp...G.l...  
0050 2e 55 50 f7 db 84 1a 9b 86 c4 f0 17 03 01 01 30 ...UP... ..0  
0060 a5 c8 27 68 0c e6 38 d2 54 c5 7e f6 0f 54 81 85 ...h..8.T...T...  
0070 89 fa 1e 02 a3 cb b0 d1 18 3a 85 77 7b 57 66 b3 ...w(Hf...  
0080 17 b6 37 54 af 45 2b 1c b4 c6 d8 a4 29 a8 9b d8 ...7T.E+...  
0090 ac 77 4d 3a 1a 56 a5 f1 8a 92 3e 95 5b 72 d3 f8 ...m.V...>.[r...  
00a0 69 7e d0 f3 67 e9 7a 10 60 5e 99 de 49 bf 72 3f ...iv...z...>.I.r?>  
00b0 0c 68 62 06 23 09 3f 16 de 6a 0e a5 00 fa 12 19 ...hb.#.?.>..
```

Secure Sockets Layer (ssl), 346 バイト



ssl.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
88	20:00:20...	192.168.1...	192.168.1...	TLSv1	320	New Session Ticket, Change Cipher Spec, ...
89	20:00:20...	192.168.1...	192.168.1...	TCP	54	10189→443 [ACK] Seq=355 Ack=780 Win=2611...
90	20:00:20...	192.168.1...	192.168.1...	HTTP	400	GET / HTTP/1.1
91	20:00:20...	192.168.1...	192.168.1...	HTTP	666	HTTP/1.1 200 OK (text/html)

> Frame 90: 400 bytes on wire (3200 bits), 400 bytes captured (3200 bits) on interface 0
> Ethernet II, Src: Inventec_2f:9c:c6 (00:8c:fa:2f:9c:c6), Dst: AvalueTe_02:0b:79 (00:04:5f:02:0b:79)
> Internet Protocol Version 4, Src: 192.168.100.122 (192.168.100.122), Dst: 192.168.100.200 (192.168.100.200)
> Transmission Control Protocol, Src Port: 10189, Dst Port: 443, Seq: 355, Ack: 780, Len: 346
> Secure Sockets Layer

[2 Reassembled SSL segments (276 bytes): #90(1), #90(275)]

```
ertext Transfer Protocol  
ET / HTTP/1.1\r\n  
Accept: text/html, application/xhtml+xml, */*\r\n  
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n  
Accept-Encoding: gzip, deflate\r\n  
DNT: 1\r\n  
Host: 192.168.100.200\r\n  
Connection: Keep-Alive\r\n  
Accept-Language: ja-en-US;q=0.7,en;q=0.3\r\n  
0000 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a GET / HT TP/1.1..  
0010 41 63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d Accept: text/htm..  
0020 6c 2c 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 l, appli cation/x..  
0030 68 74 6d 6c 2b 78 6d 6c 2c 20 2a 2f 2a 0d 0a 55 html+xml , */*..l..  
0040 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c ser-Agen t: Mozil..  
0050 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 la/5.0 ( Window..  
0060 4e 54 20 3e 2e 33 3b 20 57 4f 57 36 34 3b 20 54 NT 6.3; WOW64; T..  
0070 72 69 64 65 6e 74 2f 37 2e 30 3b 20 72 76 3a 31 rident/7 .0; rv:1
```

Frame (400 bytes) Decrypted SSL data (1 byte) Decrypted SSL data (275 bytes) Reassembled SSL (276 bytes)

Hypertext Transfer Protocol (http), 276 バイト



3. Decrypting TLS/SSL data without key pair





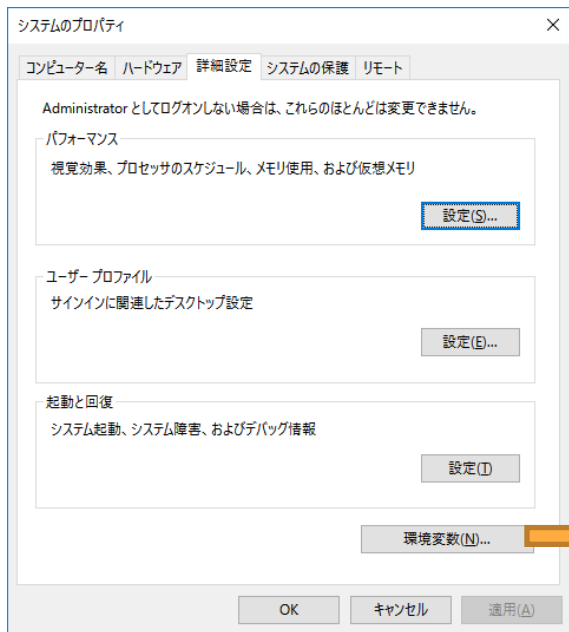
3. Decrypting TLS/SSL data without key pair

1. Set system enviromental variable
`SSLKEYLOGFILE=Path of the premaster secret`
2. Capture packets using Chrome
Check `SSLKEYLOGFILE` was generated
3. Set (Pre)-Master-Secret log filename in SSL preference

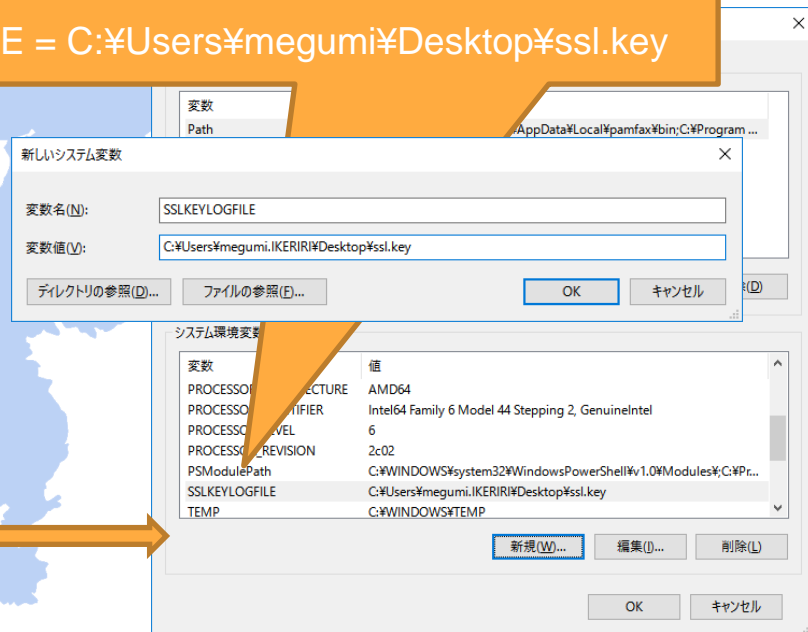


3. Decrypting TLS/SSL data without key pair

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



SSLKEYLOGFILE = C:¥Users¥megumi¥Desktop¥ssl.key





3. Decrypting TLS/SSL data without key pair

2. Capture packets using Chrome (<https://www.ikeriri.ne.jp/>) ssl2.pcapng

1. Check SSLKEYLOGFILE was generated

ssl2.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
1	16.10.43.646789	10.0.0.15	211.5.104.181	TCP	66	56521→443 [SYN] Seq=0 Win=8192 Len=0...
2	16.10.43.647407	10.0.0.15	211.5.104.181	TCP	66	56522→443 [SYN] Seq=0 Win=8192 Len=0...
3	16.10.43.648084	211.5.104.181	10.0.0.15	TCP	62	443→56521 [SYN, ACK] Seq=0 Ack=1 Win=...
4	16.10.43.648886	211.5.104.181	10.0.0.15	TCP	62	443→56522 [SYN, ACK] Seq=0 Ack=1 Win=...

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
 Ethernet II, Src: Microsoft_b7:33:99 (28:18:78:b7:33:99), Dst: Fortinet_b0:6a:9a (00:09:0f:b0:6a:9a)
 Internet Protocol Version 4, Src: 10.0.0.15 (10.0.0.15), Dst: asashina.ikeriri.ne.jp (211.5.104.181)
 Transmission Control Protocol, Src Port: 56521, Dst Port: 443, Seq: 0, Len: 0

0000 00 09 0f b0 6a 9a 28 18 78 b7 33 99 08 00 45 00 ...j.(. x.3...E.
 0010 00 34 5a ac 40 00 80 06 00 00 0a 00 0f d3 05 ...4Z.@... ..
 0020 68 b5 dc c9 01 bb df c1 2e 5f 00 00 00 80 02 h.....
 0030 20 0e 45 f0 00 00 02 04 05 b4 01 03 03 08 01 01 ..E.....
 0040 04 02 ..

ssl2.pcapng

パケットキャプチャはいけりり

https://www.ikeriri.ne.jp

パケットキャプチャはいけりり★ネットワークサービス

Wireshark 無線 有線 教育 会社情報 お問い合わせ

http contains "いけりり★ネットワークサービス株式会社 http://www.ikeriri.ne.jp/"

No.	Time	Source	Destination	Protocol	Length	Info
1	0...	192.1..	203...	DNS	77	Standard query 0xe606 A www...
2	1..	203.1..	192...	DNS	130	Standard query response 0x...

sslkey - TeraPad

```

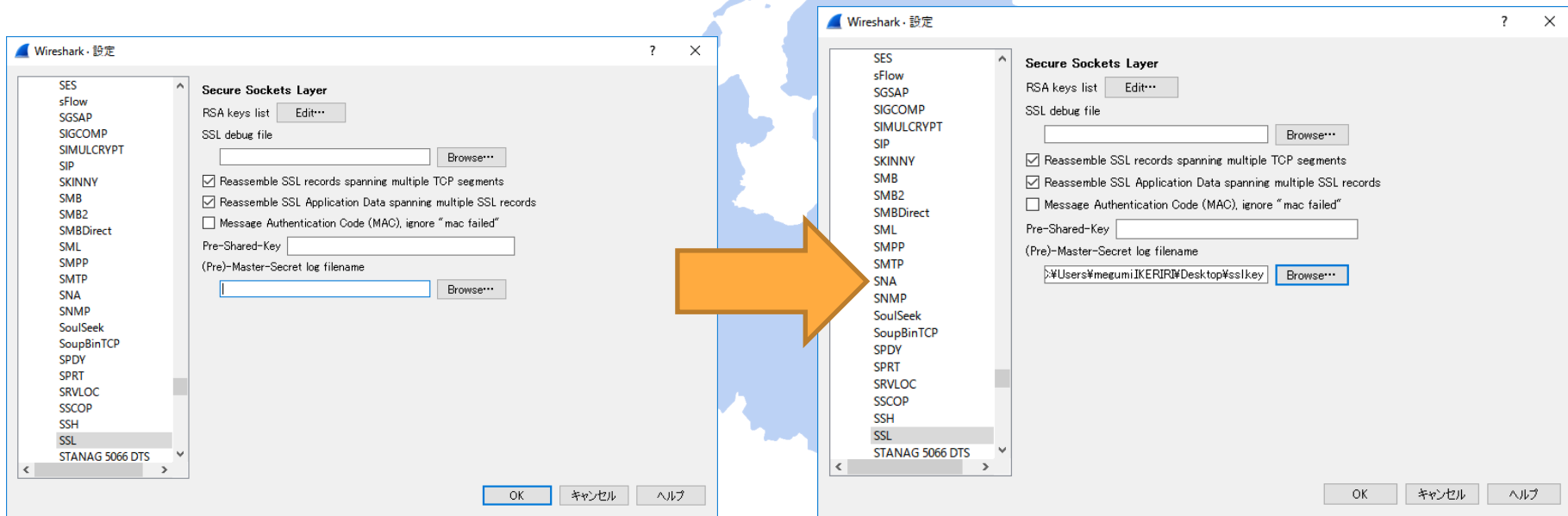
1 CLIENT_RANDOM_21bde7ccd54ab55f2a33f7e00d8a4e8db4799b3bb0923a827a651c12ebaf0c7.9
2 417428dba358dc841abc71ee870e115b9ba4141e4fc6f12d3e8158809abd92a006e2d9f396fe31
3 4ff54217829e5a6
4 CLIENT_RANDOM_11fb606c09e95e77cac0918a8a82f102f29c39ad91e0cde1d82e19df d27fe6ee.4
5 19f857f767a32769071b1057b92333afafdb7f9d455c4bd0075d98e9a21514c079b484c6fa649
6 910a9cd380788d1
7 CLIENT_RANDOM_aa3a1e3b269d704b1f472685b9e14f57e83ae9bdf d02350c0e3d6a7994f2586.9
8 ad2e2e829df120695ee47e9c02917afac9a2fa5e69108cf002432efc01beca32613f78a17e89095d
9 23af50de0b1cc7a
10 CLIENT_RANDOM_69c44dcfd4db137b695f45080f1519717751806c11012e85076d1652adc40183.0
11 9f10b50e3edc318c3abc3c3e267f35b968f01232cbaac1be8cf651d6a620bc4544fa96268b25d0
12 3ede98dda3c381c
13 CLIENT_RANDOM_014ce15eff2eb4ed48f3d1c5bce98305580559c34170932f1caba8e93a870eab.4
14 075f7a7ff69d9fa044265c0cfe53e4baf2d966f6676c431c2a75165118eadf70936c282e6a354b
15 b9b02fb527c508d
16 CLIENT_RANDOM_598e982337624cad6cb85f77b053726781c212ce0914830b3aa96f697a64828.8
17 3ccb28f78f886a319dd01f086d106e19e3ca18ef19465126e1fb4e22a1581843a3f04f7dd554cbe
18 069e1344a183f1
  
```



3. Decrypting TLS/SSL data without key pair

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

ssl.key





3. Decrypting TLS/SSL data without key pair

Check #23,28 in trace file ssl2.pcapng

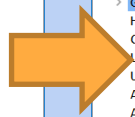
ssl2.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
22	16:10:43.719725	10.0.0.15	211.5.104.181	TCP	54	56522→443 [ACK] Seq=525 Ack=3851 Win=...
23	16:10:44.095939	10.0.0.15	211.5.104.181	TLSv1	512	Application Data, Application Data
24	16:10:44.097781	211.5.104.181	10.0.0.15	TCP	1514	[TCP segment of a reassembled PDU]
25	16:10:44.097784	211.5.104.181	10.0.0.15	TCP	1514	[TCP segment of a reassembled PDU]

Frame 23: 512 bytes on wire (4096 bits), 512 bytes captured (4096 bits) on interface 0

- Ethernet II, Src: Microsof_b7:33:99 (28:18:78:b7:33:99), Dst: Fortinet_b0:6a:9a (00:09:0f:b0:6a:9a)
- Internet Protocol Version 4, Src: 10.0.0.15 (10.0.0.15), Dst: asashina.ikeriri.ne.jp (211.5.104.181)
- Transmission Control Protocol, Src Port: 56521, Dst Port: 443, Seq: 525, Ack: 3851, Len: 458
- Secure Sockets Layer

```
0030 f7 4e 47 ae 00 00 17 03 01 00 20 bb 34 c5 12 c8 .NG... ..4...
0040 da 2c dc a2 fc 66 8e f9 9f a4 45 75 5c 93 d3 36 ....F...Eu...6
0050 5e e9 95 51 ee d4 72 7e 87 3b 93 17 03 01 01 a0 V...Q...;.....
0060 83 b0 a5 db f5 c7 07 3a 46 0f de 30 71 60 ee fe ..a.m.X.nw...V
0070 88 97 c5 5c 34 23 0b c5 75 5d df 40 ae 78 ab 4d ..X4#..u]@.x.V
0080 f7 d9 61 97 6d fe 58 5f 6e 77 b6 0f ee 56 3c ac ..a.m.X.nw...V
0090 2b da c1 59 aa 59 78 fd 7a 4e fd 22 25 68 0d 2a .Y.Yx.zN."%h.*
00a0 4e 60 bd 49 94 09 85 51 97 d6 51 8e 32 04 c6 b6 F...Q...Q.z...
00b0 4f a3 71 1c 50 e6 22 be 3b 17 80 b9 74 d8 49 0d 0.g.P"...t..t..
```



ssl2.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
22	16:10:43.719725	10.0.0.15	211.5.104.181	TCP	54	56522→443 [ACK] Seq=525 Ack=3851 Win=...
23	16:10:44.095939	10.0.0.15	211.5.104.181	HTTP	512	GET / HTTP/1.1
24	16:10:44.097781	211.5.104.181	10.0.0.15	TCP	1514	[TCP segment of a reassembled PDU]
25	16:10:44.097784	211.5.104.181	10.0.0.15	TCP	1514	[TCP segment of a reassembled PDU]

Frame 23: 512 bytes on wire (4096 bits), 512 bytes captured (4096 bits) on interface 0

- Ethernet II, Src: Microsof_b7:33:99 (28:18:78:b7:33:99), Dst: Fortinet_b0:6a:9a (00:09:0f:b0:6a:9a)
- Internet Protocol Version 4, Src: 10.0.0.15 (10.0.0.15), Dst: asashina.ikeriri.ne.jp (211.5.104.181)
- Transmission Control Protocol, Src Port: 56521, Dst Port: 443, Seq: 525, Ack: 3851, Len: 458
- Secure Sockets Layer
- [2 Reassembled SSL segments (389 bytes): #23(1), #23(388)]
- Hypertext Transfer Protocol
- GET / HTTP/1.1\r\n
- Host: www.ikeriri.ne.jp\r\n
- Connection: keep-alive\r\n
- Upgrade-Insecure-Requests: 1\r\n
- User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.14
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
- Accept-Encoding: gzip, deflate, sdch, br\r\n

```
0000 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a GET / HT TP/1.1..
0010 48 6f 73 74 3a 20 77 77 77 2e 69 6b 65 72 69 72 Host: ww w.ikeriri
0020 69 2e 6e 65 2e 6a 70 0d 0a 43 6f 6e 6e 65 63 74 i.ne.jp. .Connect
0030 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: kee p-alive.
0040 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 .Upgrade-Insecur
0050 65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 55 e-Request ts: 1..U
0060 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6d ser-Agen t: Mozil
0070 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 la/5.0 ( Windows
```



4. Retrieving Unicode Characters





4. Retrieving Unicode Characters

1. Capture packets including Unicode Website (<http://www.ikeriri.ne.jp/wireshark/cheer.html>)
2. Choose TCP packet, select Follow TCP Stream
Byte stream between client and server shows up
3. Select “UTF-8” from list box of “Show and save data as”, we can read Japanese contents !

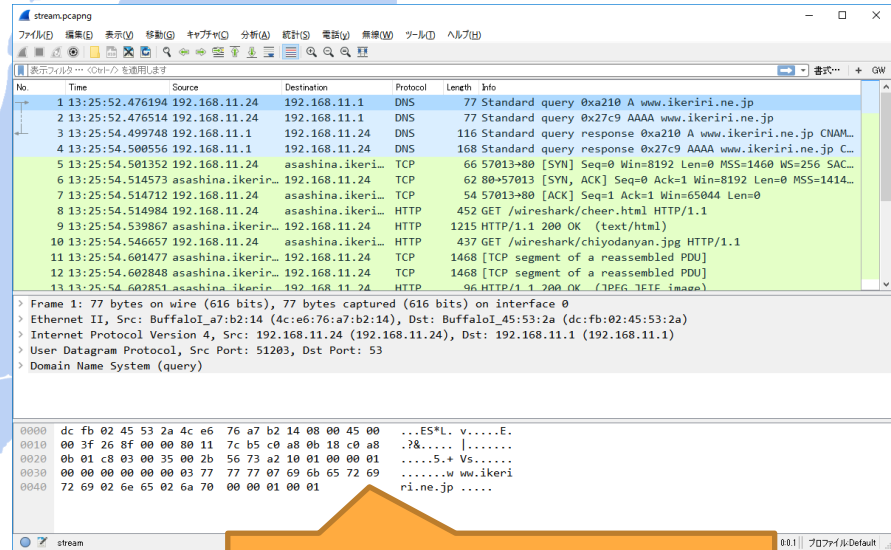


4. Retrieving Unicode Characters

1. Capture packets including Unicode Website (<http://www.ikeriri.ne.jp/wireshark/cheer.html>)



Unicode(Japanese) webpage



stream.pcapng

4. Retrieving Unicode Characters

2. Choose TCP packet, select Follow TCP Stream Byte stream between client and server shows up

Webアクセスの/パケット.pcapng

表示フィルタ: <Ctrl-F> を適用します

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.100.127	192.168.100.254	DNS	77	Standard query 0x0658 A ikeriri...
2	0.074708	192.168.100.254	192.168.100.127	DNS	116	Standard query response 0x06...
3	0.075402	192.168.100.127	211.5.104.181	TCP	66	3036→80 [SYN] Seq=0 Win=8192...
4	0.075402	211.5.104.181	192.168.100.127	TCP	62	80→3036 [SYN, ACK] Seq=0 Ack...
5	0.075402	192.168.100.127	211.5.104.181	TCP	54	3036→80 [ACK] Seq=1 Ack=1 Wi...
6	0.075402	211.5.104.181	192.168.100.127	HTTP	450	GET /cheer.html HTTP/1.1

右クリックメニュー:

- 追跡
 - TCPストリーム
 - UDPストリーム
 - SSLストリーム
 - HTTPストリーム

1. Select one of TCP packet,
2. Right click and select "Follow" from popup menu,
3. Select TCP stream from sub menu.

Wireshark - TCP ストリーム (tcp.stream eq 0) を追跡 - stream

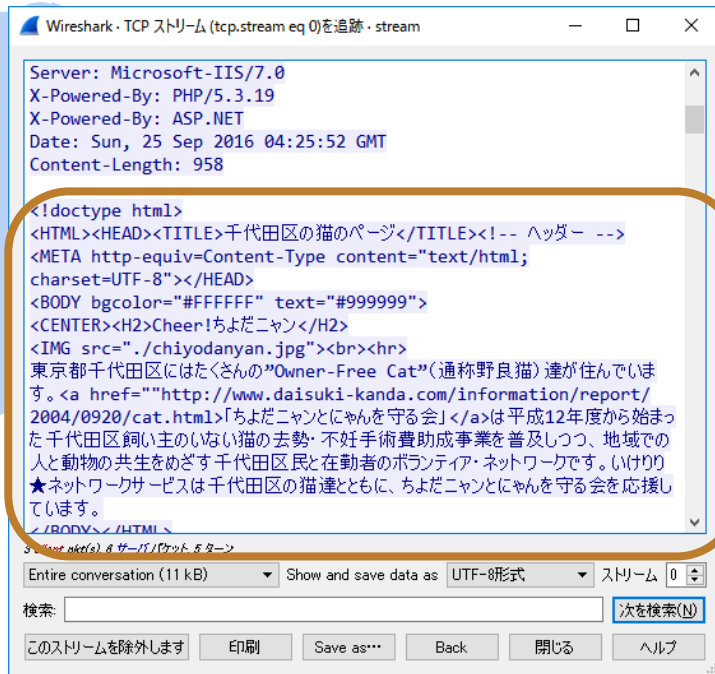
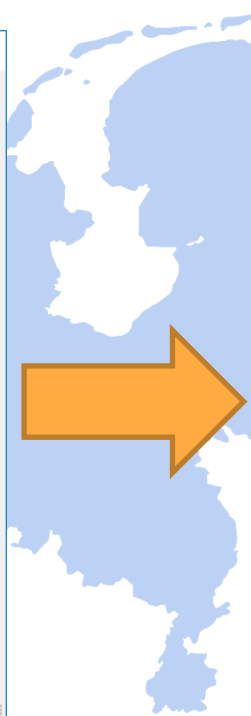
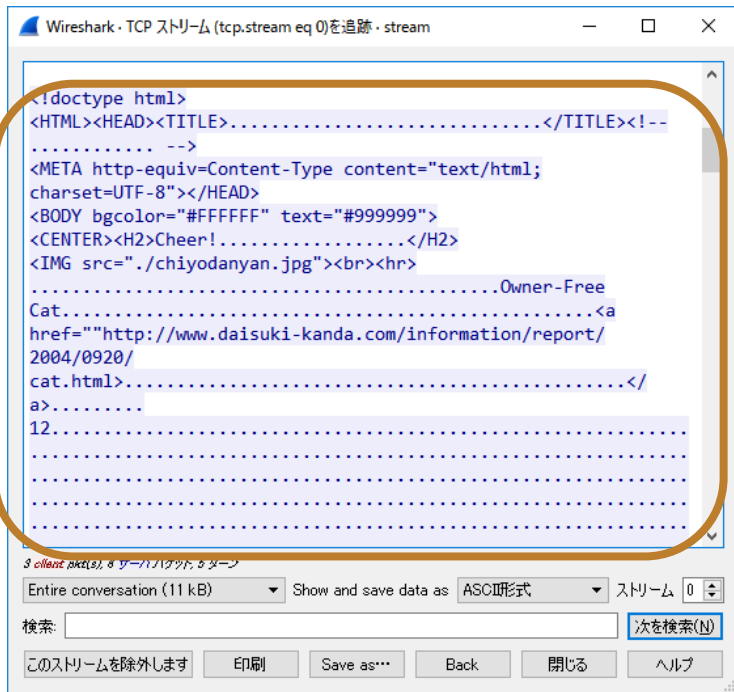
```
GET /wireshark/cheer.html HTTP/1.1
Host: www.ikeriri.ne.jp
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: ja,en-US;q=0.8,en;q=0.6

HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Type: text/html
Server: Microsoft-IIS/7.0
X-Powered-By: PHP/5.3.19
X-Powered-By: ASP.NET
Date: Sun, 25 Sep 2016 04:25:52 GMT
Content-Length: 958
```



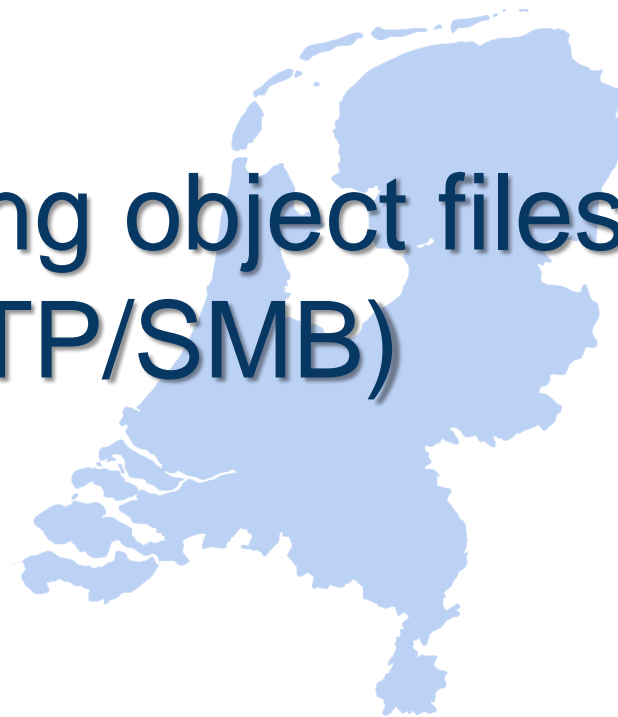
4. Retrieving Unicode Characters

3. Select “UTF-8” from list box of “Show and save data as”, we can read Japanese contents !





5. Retrieving object files (HTTP/TFTP/SMB)





5. Retrieving object files (HTTP/TFTP/SMB)

1. Capture packets including object (HTTP/TFTP/SMB) (<http://www.ikeriri.ne.jp/wireshark/cheer.html>)
note do not use cache mechanism
2. Choose File > Export Objects > HTTP (also DICOM / SMB / TFTP) and HTTP Object list shows up
3. Select “Save All” into a temporal folder and, we can retrieve CSS, ICO, JPEG, PNG, HTML, ZIP, and more.

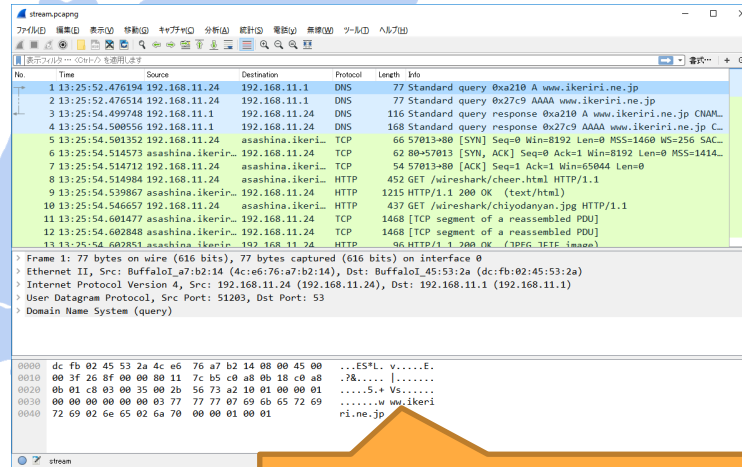


5. Retrieving object files (HTTP/TFTP/SMB)

1. Capture packets including object (HTTP/TFTP/SMB) (<http://www.ikeriri.ne.jp/wireshark/cheer.html>)
note do not use cache mechanism



Unicode(Japanese) webpage



stream.pcapng



5. Retrieving object files (HTTP/TFTP/SMB)

2. Choose File > Export Objects > HTTP (also DICOM / SMB / TFTP) and HTTP Object list shows up

The screenshot shows the 'File' menu in Wireshark. The 'Export Objects' option is selected and expanded, showing sub-options: 'DICOM...', 'HTTP...', 'SMB...', and 'TFTP...'. The 'HTTP...' option is highlighted. Below the menu, a packet list table is visible with columns: No., Time, Destination, Protocol, Length, Info. Packet 24 is highlighted in green, showing a GET request to asashina.ikeriri.ne.jp.

No.	Time	Destination	Protocol	Length	Info
24	192.168.11.1	DNS	77	Standard query query	
24	192.168.11.1	DNS	77	Standard query response	
1	192.168.11.24	DNS	116	Standard query query	
1	192.168.11.24	DNS	168	Standard query response	
24	asashina.ikeriri...	TCP	66	57013->57013	
eriri...	192.168.11.24	TCP	62	80->57013	
24	asashina.ikeriri...	TCP	54	57013->57013	
24	asashina.ikeriri...	HTTP	452	GET /www.ikeriri.ne.jp/cheer.html	
eriri...	192.168.11.24	HTTP	1215	HTTP/1.1 200 OK	
24	asashina.ikeriri...	HTTP	437	GET /www.ikeriri.ne.jp/chiyodanyan.jpg	
eriri...	192.168.11.24	TCP	1468	[TCP Seq=329212800, Win=0, Len=0]	
eriri...	192.168.11.24	TCP	1468	[TCP Seq=329212800, Win=0, Len=0]	
eriri...	192.168.11.24	HTTP	96	HTTP/1.1 200 OK	



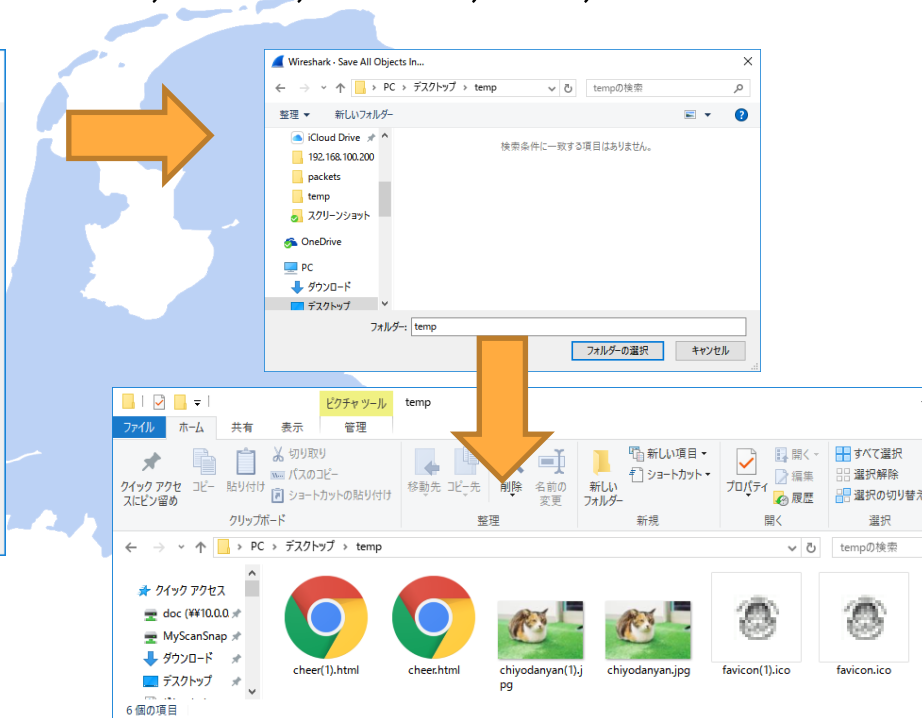
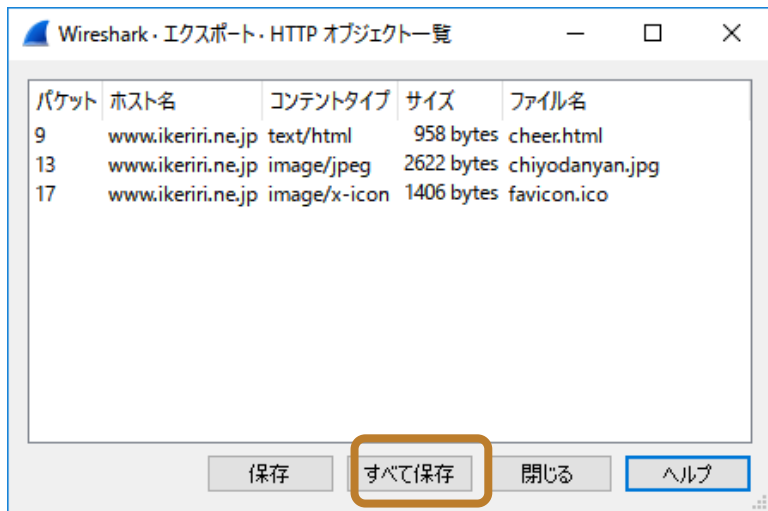
The screenshot shows the 'Wireshark - エクスポート - HTTP オブジェクト一覧' window. It displays a table of HTTP objects with columns: パケット (Packet), ホスト名 (Host Name), コンテントタイプ (Content Type), サイズ (Size), and ファイル名 (File Name). Three objects are listed: cheer.html, chiyodanyan.jpg, and favicon.ico.

パケット	ホスト名	コンテントタイプ	サイズ	ファイル名
9	www.ikeriri.ne.jp	text/html	958 bytes	cheer.html
13	www.ikeriri.ne.jp	image/jpeg	2622 bytes	chiyodanyan.jpg
17	www.ikeriri.ne.jp	image/x-icon	1406 bytes	favicon.ico



5. Retrieving object files (HTTP/TFTP/SMB)

3. Select “Save All” into a temporal folder and, we can retrieve CSS, ICO, JPEG, PNG, HTML, ZIP, and more.





6. Retrieving values of field





6. Retrieving values of field

1. If you need data values of field in the trace, tshark is the best CLI tool (including Wireshark)
Use `-D` option to check interface,
capture packet with `-i` interface
write trace file with `-w` trace file.
2. tshark has „`-T fields`“ option to get the value of field
with „`-e`“ Display filter (common with `-r` tracefile)
3. Tshark is more useful connecting another command
with redirect, pipe (Windows 10 has bash shell)



6. Retrieving values of field

1. If you need data values of field in the trace, tshark is the best CLI tool
Use `-D` option to check interface, capture packet with `-i` interface
write trace file with `-w` trace file.

Same order,
Same setting
as default
Wireshark setting,

Set `-i` interface,
`-w` trace file
and capture
packets in CLI
(`dumpcap` is
better)

```
コマンドプロンプト
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\megumi>tshark -D
1. ¥Device¥NPF_{2D82BEBE-1E21-4E44-AF3E-3F9C4EA6748C} (網、網シ、網オ、網イ、網エ、網「記
2. ¥Device¥NPF_{4F0D27B0-E075-4FF7-9D28-BA58ABD38307} (網、網シ、網オ、網イ、網エ、網「記
3. ¥Device¥NPF_{F4FFD736-AD7B-455A-9DD7-EAF47FB51F31} (網、網シ、網オ、網イ、網エ、網「記
4. ¥Device¥NPF_{B98C1496-712F-4755-AEB9-87981430B800} (網、網シ、網オ、網イ、網エ、網「記
5. ¥Device¥NPF_{411DFF04-625C-4BA3-A3DB-25EFF691C708} (Wi-Fi)
6. ¥¥. ¥USBPcap1 (USBPcap1)
7. ¥¥. ¥USBPcap2 (USBPcap2)

C:\Users\megumi>tshark -i 5 -w trace.pcapng
Capturing on 'Wi-Fi'
8
```



6. Retrieving values of field

2. tshark has „-T fields“ option to get the value of field with „-e“ Display filter (common with -r tracefile)

コマンドプロンプト

```
C:\Users\megumi\Desktop>tshark -T fields -e ip.src -e ip.dst -r stream.pcapng
192.168.11.24 192.168.11.1
192.168.11.24 192.168.11.1
192.168.11.1 192.168.11.24
192.168.11.1 192.168.11.24
192.168.11.24 211.5.104.181
211.5.104.181 192.168.11.24
192.168.11.24 211.5.104.181
192.168.11.24 211.5.104.181
211.5.104.181 192.168.11.24
192.168.11.24 211.5.104.181
211.5.104.181 192.168.11.24
```

Set Output style
As field value

Set -e Display filter string of each fields

Tshark checks Each packet and pick up
matched value (note : if there are no
matched fields, tshark outputs blank line)



6. Retrieving values of field

3. Tshark is more useful connecting another command with redirect, pipe (Windows 10 has bash shell)

```
tshark -T fields -e dns.time -r stream.pcapng | find "0" | sort
```

1:Read stream.pcapng, pick up dns response time (dns.time)

note: Not matched packet returns blank line

2:Use pipe to filter whether there are "0" string

note: time value has "0" (may be 😊))

3:Use pipe to sort ascending.

```
Result 2.023554000
```

```
2.024042000
```




7. Retrieving JSON data





7.Retrieving JSON data

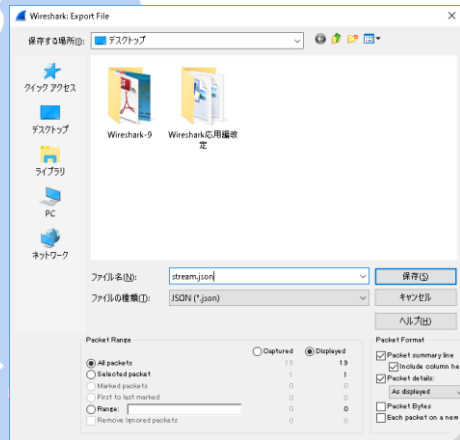
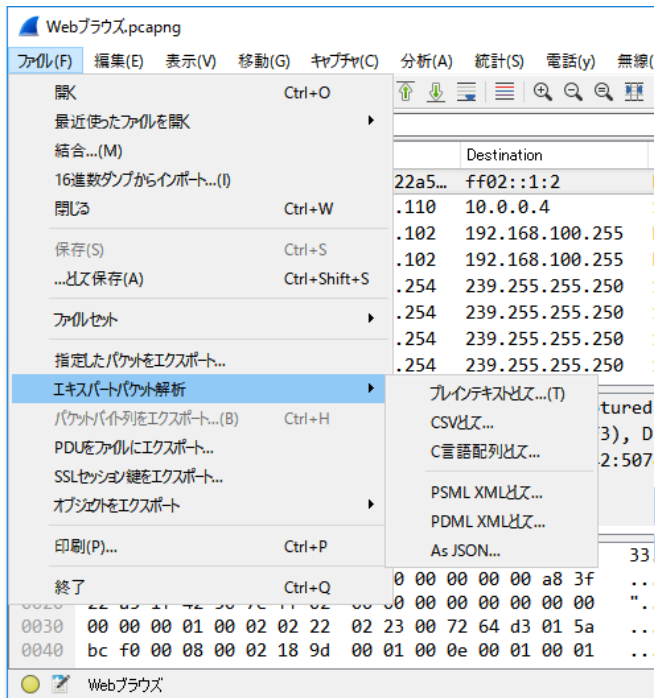
1. JSON (JavaScript Object Notation) is one of open source data format like XML, JSON is commonly used by many web application, data exchange. Wireshark can export packets as JSON. You may check JSON file by JSON Viewer.
2. Tshark can also output packets as JSON and EK(ELK) for open source data analysis tool used by Elastic Search, (Logstash,) Kibana.



7.Retrieving JSON data

1. Wireshark can export packets as JSON.

1. Select File > Expert packet analysis > As JSON
2. Set the range of packets, format of file type, and file name as XXXX.json in Export File dialog.





7.Retrieving JSON data

JSON Viewer (<http://jsonviewer.codeplex.com/>)

The screenshot shows the JSON Viewer application with a tree view on the left and a Property Grid on the right. The tree view shows a JSON array with 15 elements. The first element is an object with properties: `_index` (value: "packets-2016-10-08"), `_type` (value: "pcap_file"), `_score` (value: <null>), and `_source` (value: <null>). The `_source` property is expanded to show a `layers` array with 15 elements. The first element of the `layers` array is an object with properties: `frame`, `eth`, `ip`, `udp`, and `dns`. The Property Grid shows the following data:

Property	Value
Destination GeoIP:	Unknown
ip.addr	192.168.11.1
ip.checksum	0x00007cb5
ip.checksum.status	2
ip.dsfield	JsonObject
ip.dst	192.168.11.1
ip.dst.host	apdcfb0245532a
ip.flags	JsonObject
ip.frag.offset	0
ip.hdr.len	20
ip.host	apdcfb0245532a
ip.id	0x0000268f
ip.len	63
ip.proto	17
ip.src	192.168.11.24
ip.src.host	192.168.11.24
ip.ttl	128
ip.version	4
Source GeoIP:	Unknown

„JSON Viewer,, is an open source JSON viewer and editor application on Windows.

The screenshot shows the JSON Viewer application with the raw JSON data displayed in a text editor. The JSON data is as follows:

```
{
  "_index": "packets-2016-10-08",
  "_type": "pcap_file",
  "_score": null,
  "_source": {
    "layers": {
      "frame": {
        "frame.interface_id": "0",
        "frame.encap_type": "1",
        "frame.time": "Sep 25, 2016 13:25:52.476194000 Wu00147Wu00140Wu00139Wu00158 (Wu00149Wu00143Wu00128Wu00142Wu00150)",
        "frame.offset_shift": "0.000000000",
        "frame.time_epoch": "147477552.476194000",
        "frame.time_delta": "0.000000000",
        "frame.time_delta_displayed": "0.000000000",
        "frame.time_relative": "0.000000000",
        "frame.number": "1",
        "frame.len": "77",
        "frame.cap_len": "77",
        "frame.marked": "0",
        "frame.ignored": "0",
        "frame.protocols": "ethertype:ip:udp:dns",
        "frame.coloring_rule_name": "UDP"
      }
    }
  }
}
```



7.Retrieving JSON data

2. Tshark can also output packets as JSON and EK(ELK) for open source data analysis tool used by Elastic Search, (Logstash,) Kibana.

```
tshark -T (json|ek) -r stream.pcapng > temp.json
```

1: Read stream.pcapng, convert JSON | EK

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

コマンドプロンプト

```
C:\Users\megumi\Desktop>tshark -T json -r stream.pcapng > temp.json
```

```
C:\Users\megumi\Desktop>tshark -T ek -r stream.pcapng > temp2.json
```



8. Wireshark is the source of big data analysis !





8. Wireshark is the source of big data analysis !

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 !

```
64 80 99 0a a5 e8 dc fb 02 45 5:  
00 28 15 fd 40 00 75 06 e8 52 d:  
0b 1d 01 bb 26 c6 fd e1 b4 db 4:  
fe 14 1b 0a 00 00
```

Live packet



Decode / dissection



Big data analysis
Full-text search



Kibana

Visualize
Real-time analysis

We can recode everything in network using Wireshark, and export dissection result as JSON, JSON connect Wireshark with big data analysis.



8. Wireshark is the source of big data analysis !

```
64 80 99 0a a5 e8 dc fb 02 45 51  
00 28 15 fd 40 00 75 06 e8 52 d1  
0b 1d 01 bb 26 c6 fd e1 b4 db 44  
fe 14 1b 0a 00 00
```

Live packet



Decode / dissection



elastic +

Big data analysis
Full-text search



Kibana

Visualize
Real-time analysis

Elasticsearch is a popular open source full-text search engine based on Apache Lucene, Elasticsearch uses schema-free JSON documents. Kibana is a real-time data visualization platform, a plugin of Elasticsearch. Elasticsearch with Kibana is one of the best open source big data analysis with Wireshark JSON file.

Elasticsearch <https://www.elastic.co/products/elasticsearch>

Kibana <https://www.elastic.co/products/kibana>



8. Wireshark is the source of big data analysis !



elastic +



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 !



8. Wireshark is the source of big data analysis !

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

JDK(Java Developer Kit) 8 u101

Curl (web access command)

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

Download jdk-8u101-windows-x64.exe
Execute and start setup program

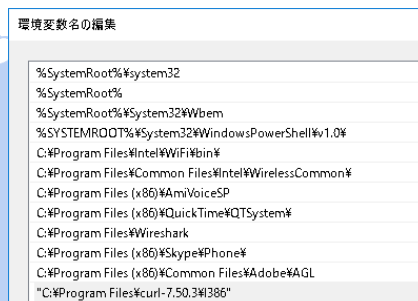
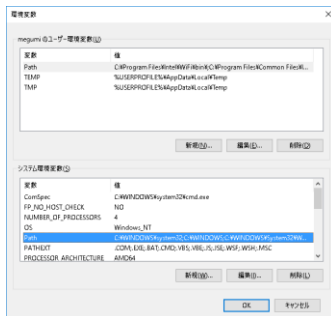
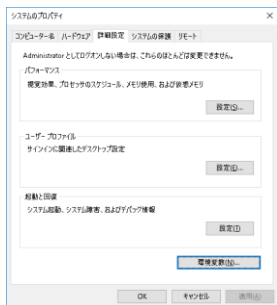
<https://curl.haxx.se/download.html>

Download curl-7.50.3.cab
Extract cab file and copy into "Program Files"
C:\Program Files\curl-7.50.3\curl.exe



8. Wireshark is the source of big data analysis !

1. Download JDK, Curl and install, and set system environmental 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¥1386

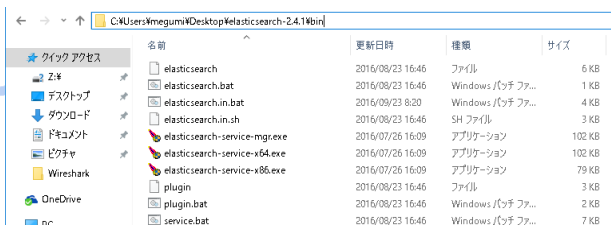
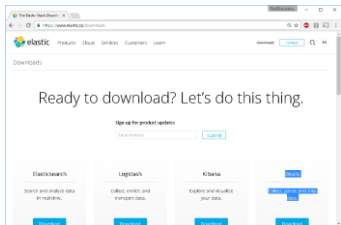
```
C:¥Users¥megumi>java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

```
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 SSI Kerberos SPNEGO NTLM SSL libz
```



8. Wireshark is the source of big data analysis !

2. Download Elasticsearch and start server, check <http://localhost:9200>



<https://www.elastic.co/downloads>

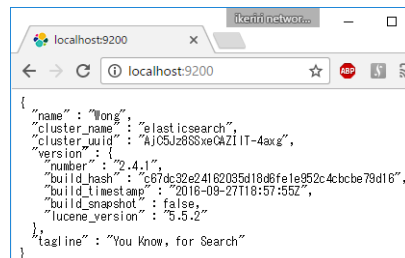
C:\Users\megumi\Desktop\elasticsearch-2.4.1\bin\elasticsearch.bat

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>

```

Elasticsearch 2.4.1
[2016-10-09 12:33:55.893] [INFO] [node
pid[8592], bui ld[67dc32/2016-09-27T18:57:55Z]
[2016-10-09 12:33:55.893] [INFO] [node
[2016-10-09 12:33:56.904] [INFO] [plugins
x, lang-expression, lang-groovy], plugins [], sites []
[2016-10-09 12:33:56.941] [INFO] [env
paths, mounts [{"C:"}], net usable_space [119.3gb], net total_space [211.5gb], spins? [unkn
own], types [NFS]]
[2016-10-09 12:33:56.941] [INFO] [env
7mb], compressed_ordinary_object_pointers [true]
[2016-10-09 12:33:59.877] [INFO] [node
[2016-10-09 12:33:59.877] [INFO] [node
[2016-10-09 12:34:00.616] [INFO] [transport
[127.0.0.1:9200], bound_addresses [127.0.0.1:9200], [{":1}:9200]
[2016-10-09 12:34:00.623] [INFO] [discovery
hDgcxLGRjOZtStgEKmg] [Spirit of '76] elasticsearch/e
[2016-10-09 12:34:04.739] [INFO] [cluster.service
[2016-10-09 12:34:04.807] [INFO] [gateway
[2016-10-09 12:34:05.191] [INFO] [http
[127.0.0.1:9200], bound_addresses [127.0.0.1:9200], [{":1}:9200]
[2016-10-09 12:34:05.191] [INFO] [node

```





8. Wireshark is the source of big data analysis !

3. Convert packet dissection data into Elasticsearch friendly JSON file

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

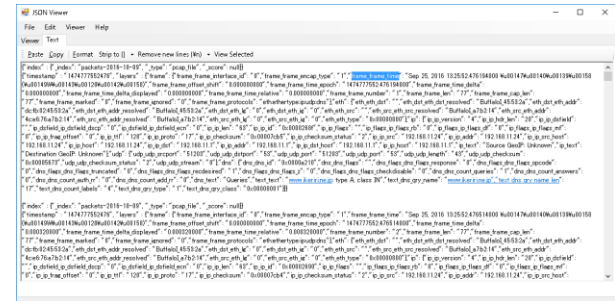
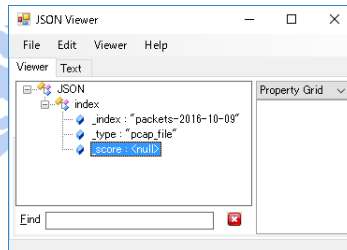
1: Read stream.pcapng, convert EK (Elasticsearch friendly JSON file)

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

コマンドプロンプト

```
C:\Users\megumi\Desktop>tshark -T ek -r stream.pcapng > packet.json
```

```
C:\Users\megumi\Desktop>_
```





8. Wireshark is the source of big data analysis !

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

```
コマンドプロンプト
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-Cj","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}
C:\Users\megumi\Desktop>
```

```
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-Cj","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},...
,{"create":{"_index":"packets-2016-10-09","_type":"pcap_file","_id":"AVenqQppWAnL3ZFeC-C0","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}
```




8. Wireshark is the source of big data analysis !

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

```

localhost:9200/_search?pretty
localhost:9200/_search?pretty

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": "AVenqPpqWANL3ZFeC-Ck",
      "_score": 1.0,
      "_source": {
        "timestamp": "1474777554499",
        "layers": {
          "frame": {
            "frame_interface_id": "0",
            "frame_encap_type": "1",
            "frame_line": "Sep 25, 2016 13:25:54.49748000 Wu00147Wu00140",
            "frame_offset_shift": "0.00000000",
            "frame_line_epoch": "1474777554.49748000",
            "frame_line_delta": "2.023234000",
            "frame_line_delta_displayed": "2.023234000",
            "frame_line_relative": "2.023554000",
            "frame_number": "3",
            "frame_len": "118",
            "frame_cap_len": "118",
            "frame_marked": "0",
            "frame_ignored": "0",
            "frame_protocols": "eth:ethertype:ip:udp:dns"
          }
        },
        "eth": {
          "eth_dst": "",
          "eth_dst_eth_dst_resolved": "Buffalo1_a7:b2:14",
          "eth_dst_eth_addr": "4c:e6:78:a7:b2:14",
          "eth_dst_eth_addr_resolved": "Buffalo1_a7:b2:14",
          "eth_dst_eth_is": "0",
          "eth_dst_eth_is_resolved": "0",
          "eth_src": "",
          "eth_src_eth_src_resolved": "Buffalo1_45:53:2a",
          "eth_src_eth_addr": "dc:fb:02:45:53:2a",
          "eth_src_eth_addr_resolved": "Buffalo1_45:53:2a",
          "eth_src_eth_is": "0",
          "eth_src_eth_is_resolved": "0"
        }
      }
    }
  ]
}

```

```

{
  "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": "AVenqPpqWANL3ZFeC-Ck",
      "_score": 1.0,
      "_source": {
        "timestamp": "1474777554499",

```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.11.24	192.168.11.1	DNS	77	Standard query 0xa210 A www.ikeriri.n...
2	0.000320	192.168.11.24	192.168.11.1	DNS	77	Standard query 0x27c9 AAAA www.ikeriri.n...
3	0.023234	192.168.11.1	192.168.11.24	DNS	116	Standard query response 0xa210 A www.ikeriri.n...
4	0.000808	192.168.11.1	192.168.11.24	DNS	168	Standard query response 0x27c9 AAAA www.ikeriri.n...
5	0.000796	192.168.11.24	asashina.ikeriri.ne.jp	TCP	66	57013->80 [SYN] Seq=0 Win=8192 Len=0 M...
6	0.013221	asashina.ikeriri.ne.jp	192.168.11.24	TCP	62	80->57013 [SYN, ACK] Seq=1 Ack=1 Win=8...
7	0.000139	192.168.11.24	asashina.ikeriri.ne.jp	TCP	54	57013->80 [ACK] Seq=1 Ack=1 Win=65044 ...
8	0.000272	192.168.11.24	asashina.ikeriri.ne.jp	HTTP	452	GET /wireshark/cheer.html HTTP/1.1
9	0.024883	asashina.ikeriri.ne.jp	192.168.11.24	HTTP	1215	HTTP/1.1 200 OK (text/html)
10	0.006790	192.168.11.24	asashina.ikeriri.ne.jp	HTTP	437	GET /wireshark/chiyodanyan.jpg HTTP/1...
11	0.054820	asashina.ikeriri.ne.jp	192.168.11.24	TCP	1468	[TCP segment of a reassembled PDU]
12	0.001371	asashina.ikeriri.ne.jp	192.168.11.24	TCP	1468	[TCP segment of a reassembled PDU]
13	0.000003	asashina.ikeriri.ne.jp	192.168.11.24	HTTP	96	HTTP/1.1 200 OK (JPEG JFIF image)
14	0.000115	192.168.11.24	asashina.ikeriri.ne.jp	TCP	54	57013->80 [ACK] Seq=782 Ack=4032 Win=6...
15	0.051055	192.168.11.24	asashina.ikeriri.ne.jp	HTTP	398	GET /favicon.ico HTTP/1.1
16	0.016414	asashina.ikeriri.ne.jp	192.168.11.24	TCP	1468	[TCP segment of a reassembled PDU]
17	0.001206	asashina.ikeriri.ne.jp	192.168.11.24	HTTP	296	HTTP/1.1 200 OK (image/x-icon)
18	0.000127	192.168.11.24	asashina.ikeriri.ne.jp	TCP	54	57013->80 [ACK] Seq=1126 Ack=5688 Win=...
19	1.496935	192.168.11.1	192.168.11.24	TCP	74	32830->80 [SYN] Seq=0 Win=5840 Len=0 M...



8. Wireshark is the source of big data analysis !

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

```
{
  "_index": "packets-2016-10-09",
  "_type": "pcap_file",
  "_id": "AVenqQpqWAnL3ZFeC-Ck",
  "_score": 1.0,
  "_source": {
    "timestamp": "1474777554499",
    "layers": {
      "frame": {
        "frame_frame_interface_id": "0",
        "frame_frame_encap_type": "1",
        "frame_frame_time": "Sep 25, 2016 13:25:54.499748000
        ¥u00147¥u00140¥u00139¥u00158
        (¥u00149W¥u00143¥u00128¥u00142¥u00158)",
        "frame_frame_offset_shift": "0.000000000",
        "frame_frame_time_epoch": "1474777554.499748000",
        "frame_frame_time_delta": "2.023234000",
        "frame_frame_time_delta_displayed": "2.023234000",
        "frame_frame_time_relative": "2.023554000",
        "frame_frame_number": "3",
        "frame_frame_len": "116",
        "frame_frame_cap_len": "116",
        "frame_frame_marked": "0",
        "frame_frame_ignored": "0",
        "frame_frame_protocols": "eth:ethertype:ip:udp:dns"
      },
      {"eth": {
        "eth_eth_dst": "",
        "eth_dst_eth_dst_resolved": "Buffalo_l_a7:b2:14",
        "eth_dst_eth_addr": "4c:e6:76:a7:b2:14",
        "eth_dst_eth_addr_resolved": "Buffalo_l_a7:b2:14",
        "eth_dst_eth_lg": "0",
        "eth_dst_eth_ig": "0",
        "eth_eth_src": "",
        "eth_src_eth_src_resolved": "Buffalo_l_45:53:2a",
        "eth_src_eth_addr": "dc:fb:02:45:53:2a",
        "eth_src_eth_addr_resolved": "Buffalo_l_45:53:2a",
        "eth_src_eth_lg": "0",
        "eth_src_eth_ig": "0",
        "eth_eth_type": "0x00000800"
      }},
      "ip": {
        "ip_ip_version": "4",
        "ip_ip_hdr_len": "20",
        "ip_ip_dsfield": "",
        "ip_dsfield_ip_dsfield_dscp": "0",
        "ip_dsfield_ip_dsfield_ecn": "0",
        "ip_ip_len": "102",
        "ip_ip_id": "0x00000000",
      }
    }
  }
}
```



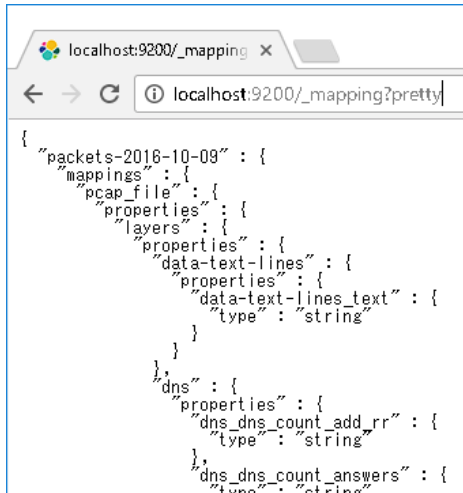

8. Wireshark is the source of big data analysis !

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

"timestamp" : {
"type" : "string"

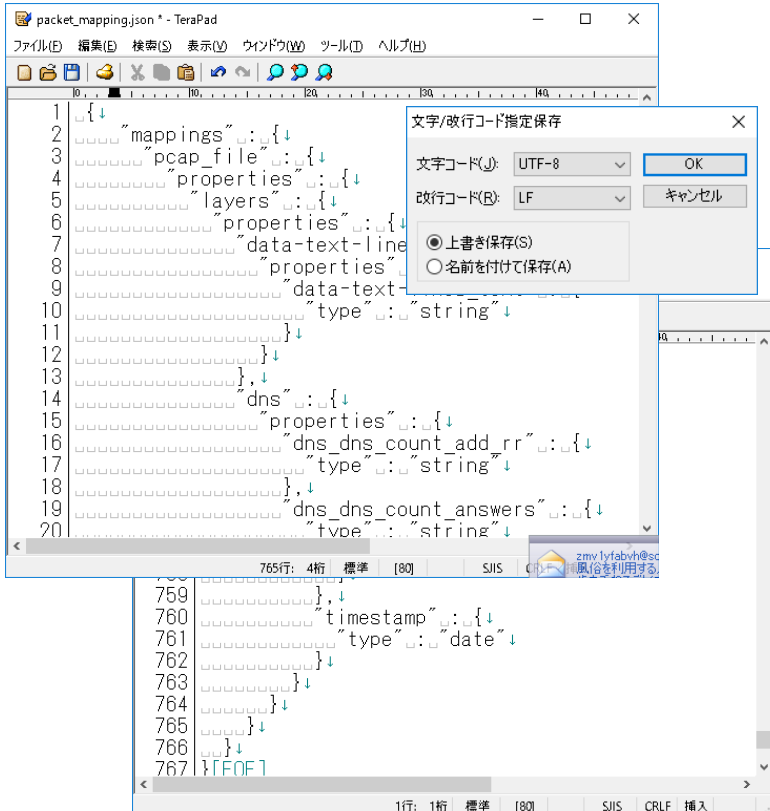


"timestamp" : "1474777554499",



8. Wireshark is the source of big data analysis !

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`”

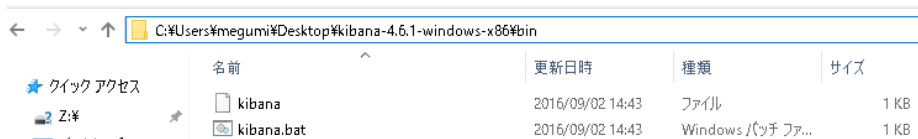
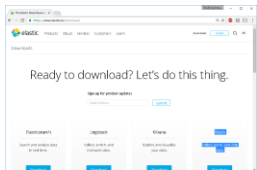
Re-enter original data

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



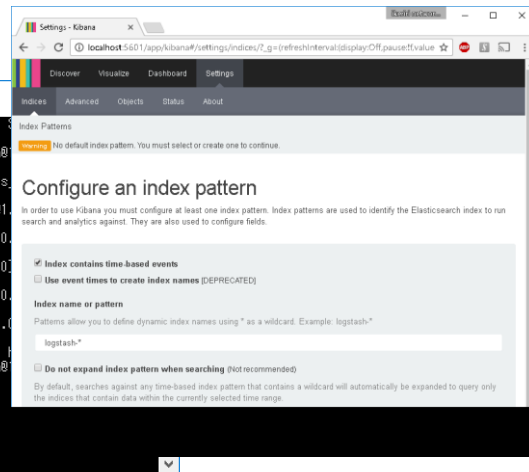
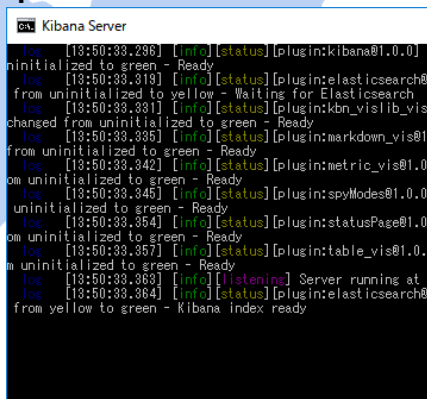
8. Wireshark is the source of big data analysis !

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>





8. Wireshark is the source of big data analysis !

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

The screenshot shows the Kibana interface for the 'packets-2016-10-09' index pattern. It displays a list of fields with their core types and control options.

name	type	format	analyzed	indexed	controls
layers.tcp.tcp_len	string		✓	✓	☑
layers.tcp.segments_tcp_reassembled_data	string		✓	✓	☑
layers.dns.text_dns_resp_ttl	string		✓	✓	☑

Configure an index pattern

In order to use Kibana you must configure at least one index pattern search and analytics against. They are also used to configure fields.

- Index contains time-based events
- Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard

packets-2016-10-09

Time-field name [refresh fields](#)

timestamp

Create



8. Wireshark is the source of big data analysis !

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

The screenshot shows the Kibana Discover interface. At the top, there's a search bar with the query `layers.ip.ip_dst`. Below the search bar, there's a message: "No results found ☹️". A message below that says: "Unfortunately I could not find any results matching your search. I tried really hard. I looked all over the place and frankly, I just couldn't find anything good. Help me, help you. Here are some ideas: Expand your time range". The interface includes a "Quick" section with time range pickers (Today, Yesterday, Last 15 minutes, etc.), a "Relative" section, and an "Available Fields" list on the left. The main area shows a list of search results with columns for time, source, and _source.

1. At First, use time picker to select the time of packets (just use “Last 1 year” is a good way)
2. Check histogram 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”

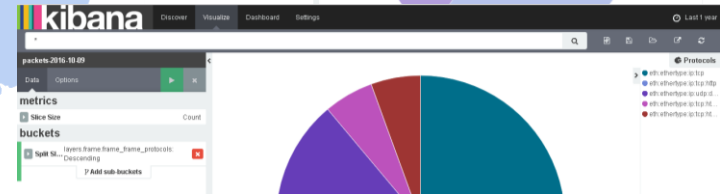
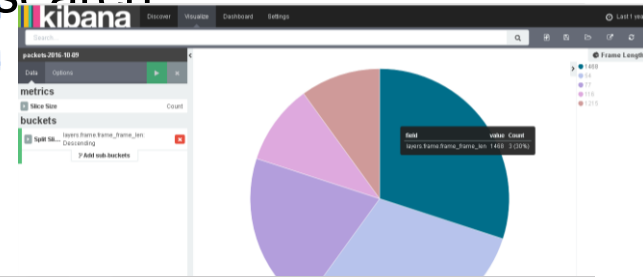
The screenshot shows the Kibana Visualize interface. The visualization is a histogram titled "layers.ip.ip_dst". The "Quick Count" shows 19/19 records. The histogram has three bars representing different IP destinations: 192.168.11.24 (52.8%), 211.5.104.181 (38.8%), and 192.168.11.1 (10.5%). The interface includes a "Visualize (1 warning)" button at the bottom. The left pane shows the "layers.ip.ip_dst" field selected. The right pane shows the "metrics" and "buckets" sections.



8. Wireshark is the source of big data analysis !

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 Aggrigation 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

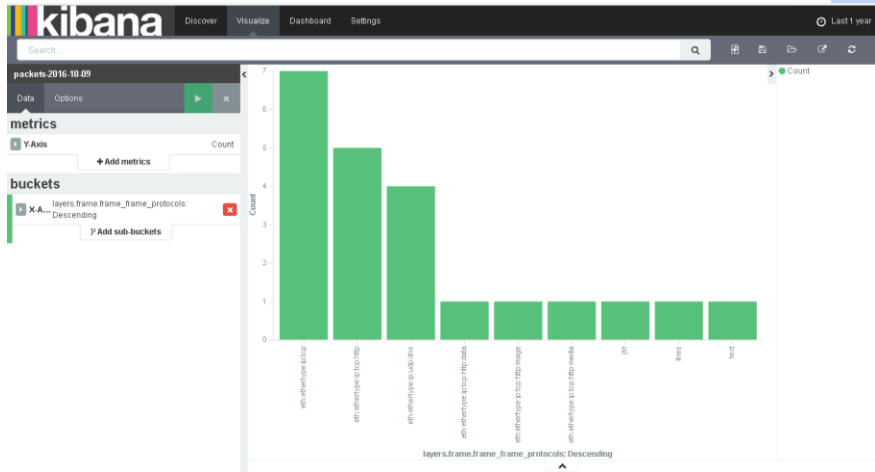




8. Wireshark is the source of big data analysis !

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 !!



 THANK YOU VERY MUCH FOR LISTENING

Use Wireshark for everything !

Thank you !

どうもありがとうございます !

