Episode 27 (Security VI: Contraption)　　Packet Analysis II

Using WireShark

Tanuki, the next step is packet analysis using WireShark.
The subject for analysis is the chat program created in episodes 6 and 7. We will use Wireshark to analyze the packets during communication between the chat in the server configuration and the chat in the client configuration.
The ASCII code will also be needed, so I will leave it as presented again.

文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進
NUL	0	00	DLE	16	10	SP	32	20	0	48	30	@	64	40	P	80	50	`	96	60	p	112	70
SOH	1	01	DC1	17	11	!	33	21	1	49	31	A	65	41	Q	81	51	a	97	61	q	113	71
STX	2	02	DC2	18	12	"	34	22	2	50	32	B	66	42	R	82	52	b	98	62	r	114	72
ETX	3	03	DC3	19	13	#	35	23	3	51	33	C	67	43	S	83	53	c	99	63	s	115	73
EOT	4	04	DC4	20	14	$	36	24	4	52	34	D	68	44	T	84	54	d	100	64	t	116	74
ENQ	5	05	NAK	21	15	%	37	25	5	53	35	E	69	45	U	85	55	e	101	65	u	117	75
ACK	6	06	SYN	22	16	&	38	26	6	54	36	F	70	46	V	86	56	f	102	66	v	118	76
BEL	7	07	ETB	23	17	'	39	27	7	55	37	G	71	47	W	87	57	g	103	67	w	119	77
BS	8	08	CAN	24	18	(	40	28	8	56	38	H	72	48	X	88	58	h	104	68	x	120	78
HT	9	09	EM	25	19	)	41	29	9	57	39	I	73	49	Y	89	59	i	105	69	y	121	79
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	J	74	4a	Z	90	5a	j	106	6a	z	122	7a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	K	75	4b	[	91	5b	k	107	6b	{	123	7b
FF*	12	0c	FS	28	1c	,	44	2c	<	60	3c	L	76	4c	¥	92	5c	l	108	6c	|	124	7c
CR	13	0d	GS	29	1d	-	45	2d	=	61	3d	M	77	4d	]	93	5d	m	109	6d	}	125	7d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	^	94	5e	n	110	6e	~	126	7e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	_	95	5f	o	111	6f	DEL	127	7f

Character	decimal	hexadecimal

From the IT Dictionary

You have studied tcpdump, so you know that when data flows over a communication line, it is divided into packets and sent. Here's a refresher.
A packet is made up of data, a header in the upper application layer (e.g., HTTP), a header in the transport layer (e.g., TCP), a header in the network layer (e.g., IP), and a header in the individual network layer (e.g., Ethernet).
In the figure, the following is shown.

← aspect of a packet flowing through a packet

Ethernet header	IP header	TCP header	Part of data	FCS
14 bytes	20 bytes	20 bytes (UDP：8 bytes)		4 bytes trailer

※FCS(trailer (vehicle)): Checks to see if packets were not corrupted during transmission.

When packets flow from PC (A) to PC (B), capturing those packets is called packet capture, right? Also, looking inside the captured packets is called packet monitoring. Needless to say, seeing and analyzing the actual packets flowing through the system is more useful than learning from a book. But if you don't learn how to operate and view the packet monitoring, it's like a cat with a panda!
As I said before, WireShark is installed on "kali Linux", so you can use it right away.

What if WireShark is not installed on anything other than "kali Linux"? For example, what if I want to use it on CentOS7 which I have built?

It is possible to install on CentOS7, but it is quite a difficult task. It would be faster to install "kali Linux", but just in case, here are the steps to install it on CentOS7.

[Installing Wireshark]

① First things first: Preparation

```
$ su – root      # become root
# mkdir tool     # Create a directory tool as root for storage
```

② Next, you need Openssl

Make sure Openssl is installed on CentOS.

```
# openssl version
```

If installed, do the following just in case

```
# yum clean all
# yum list updates
# yum update openssl
```

(If you are installing Openssl anew)

Dependency Installation

```
# yum install -y zlib-devel perl-core make gcc
```

Download openssl-1.1.1 source

```
# curl https://www.openssl.org/source/openssl-1.1.1.tar.gz
-o /usr/local/src/openssl-1.1.1.tar.gz
```

Installing openssl-1.1.1

```
# cd /usr/local/src
# tar xvzf openssl-1.1.1.tar.gz
# openssl-1.1.1/
# ./config --prefix=/usr/local/openssl-1.1.1 shared zlib
# make depend
# make
# make test
# make install
```

Installation Confirmation

```
# ls -l /usr/local/openssl-1.1.1
```

operation check

```
# /usr/local/openssl-1.1.1/bin/openssl ciphers -v TLSv1.3
```

Return to Root's current directory.

```
# cd ~
```

③Whether Openssl is already installed or a new installation, there are common tasks that must be performed.

```
# yum install openssl-devel
# yum install libgcrypt-devel
```

④ python3 is required

```
# yum install python3
```

⑤ cmake is required

Download Site

https://cmake.org/download/

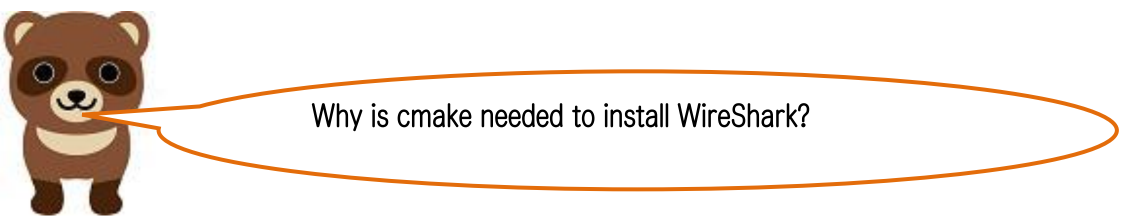
The site displays the following

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.16.3.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.16.3.zip

Binary distributions:

Save the file as /root/tool/cmake-3.16.3.tar.gz

cd tool
defrosting
tar xvfz ./cmake-3.15.3.tar.gz
Go to the unzipped directory
cd ./cmake-3.15.3
Install required packages
yum install -y gcc gcc-c++
Create makefile in bootstrap
./bootstrap
make & make install
make
make install
confirmation
cmake --version



Since WireShark is obtained from a source file, it must be built (including compilation). For this, cmake is necessary.
Now it's time to download and install WireShark.

⑥ Installing WireShark

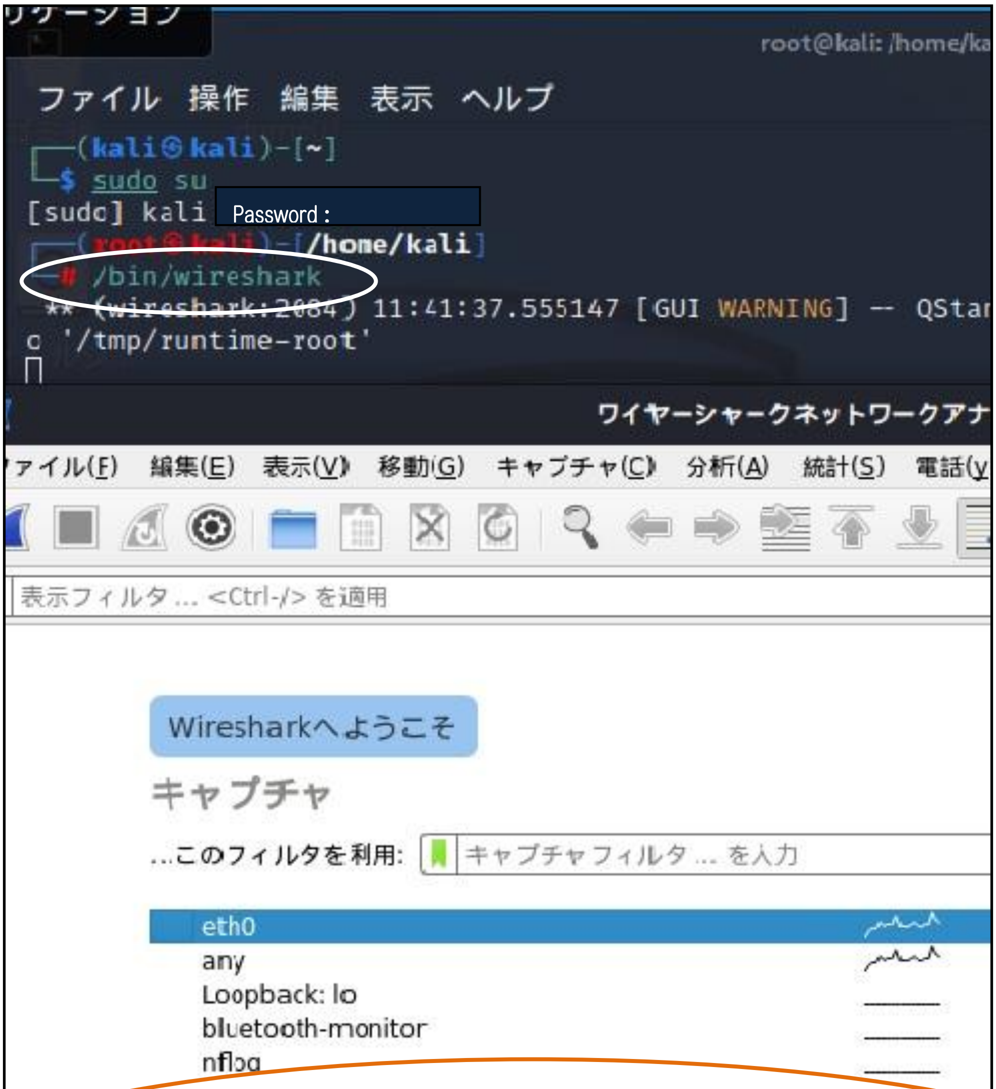
Download Site

https://www.wireshark.org/download.html

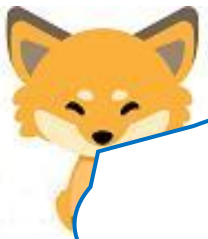
Save the file as /root/tool/wireshark-3.2.1.tar.xz

```
# cd tool
defrosting
# tar xJfv ./wireshark-3.0.5.tar.xz
Create a directory for build and enter
# mkdir build
# cd build
Install required packages
# yum install -y gcc gcc-c++ glib2-devel libgcrypt-devel flex-devel byacc  libpcap-devel qt5-qtbase-devel qt5-linguist qt5-qtmultimedia-devel qt5-qtsvg-devel
Execution of cmake
# cmake ../wireshark-3.0.5
make & install
# make
# make install
confirmation
# tshark --version
If TShark (Wireshark) 3.0.5 is displayed, you are done.
```

Now that I can use the WireShark tool, I'll try to do some packet analysis from the web server, as well as how to operate it. But I'll have to start WireShark with administrator privileges (root) as shown below to get the complete packets for analysis.
Clicking on the WireShark icon won't give me administrator privileges!

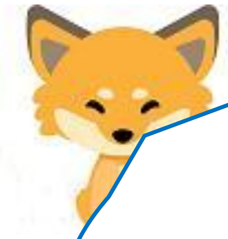


I can read it! I see, you start WireShark after it is Rooted at the terminal. So eth0 is the device name of the NIC. That's what you specify. It looks like the packet has already arrived. But it seems to be difficult if I don't get used to the operation.



Kitsune, I will explain the necessary operations for packet analysis, but you should study the various operations of WireShark by yourself by referring to the Internet or books.
As I have said before, the purpose of this site is not to teach you how to use the tools and applications.

Roger, if you have the ability to read a book, you can understand how to operate it.



In order to analyze packets using WireShark, it is better to use a demo model for easy-to-understand testing, where the contents of the transmission and reception are known in advance. So, we will use the chat programs created in episodes 6 and 7. Using the server chat program and the client chat program is the best choice since the content of the transmission is known in advance and they are intercommunicating with each other.

First, let's analyze the packets using the chat program. The chat program is running on a single PC.

[Execution state of the chat program for packet analysis.]

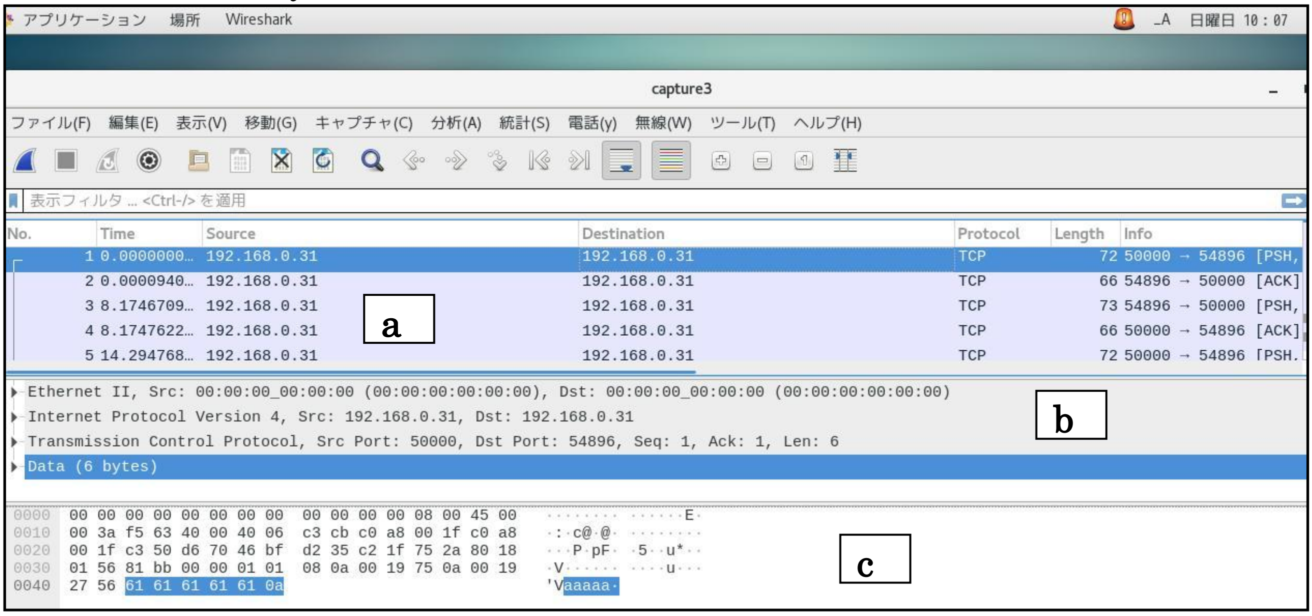
client process	server process
IP: 192.168.0.31	IP: 192.168.0.31

```
[                    ~]$ ./cctest
wait
aaaaa
Your turn.:   bbbbbb
Waiting!
ccccc
Your turn.:   ddddd
Waiting!
quit
Your turn.:   quit
Waiting!
```

```
[                    ~]$ ./sctest
Go ahead!
Your turn.:   aaaaa
Waiting!
bbbbbb
Your turn.:   ccccc
Waiting!
ddddd
Your turn.:   quit
Waiting!
quit
```

The packet analysis by WireShark is shown below.
It is important to understand the meaning of the "a," "b," and "c" parts. If you can't do that, hacking is a dream come true. Please try your best.

［Basic Packet Analysis with Wireshark］

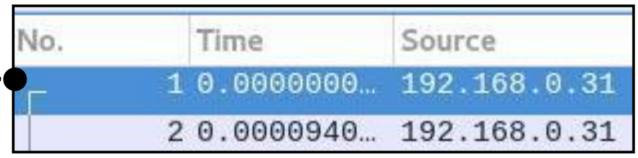


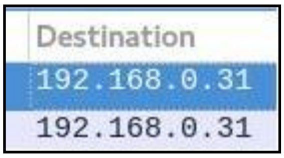
The above is divided into three parts: [a], [b], and [c].

［a］：One line represents one packet.

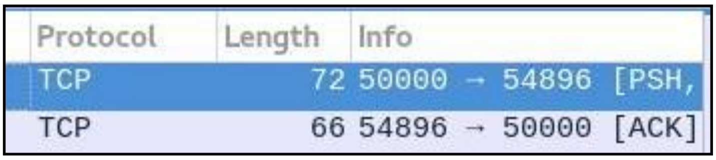
Order in which packets are sent:Elapsed time (initially 0):Sending IP Address receiving IP address

(Server process) (Client process)



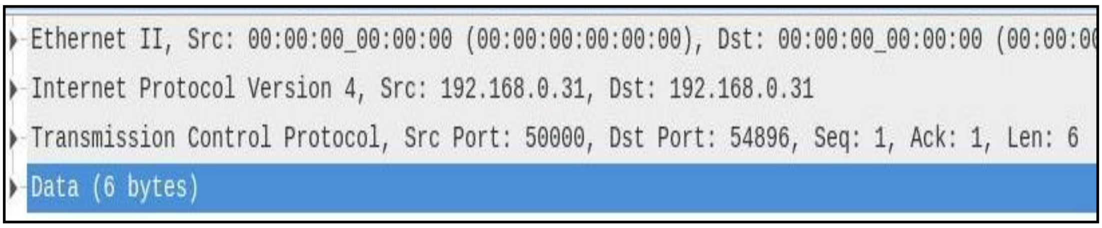


Protocol used: Frame length (bytes): Sending port (50000) to receiving port (54896)



Communication Control Flags
PSH (push): Prepare to send
ACK: Response confirmation
:

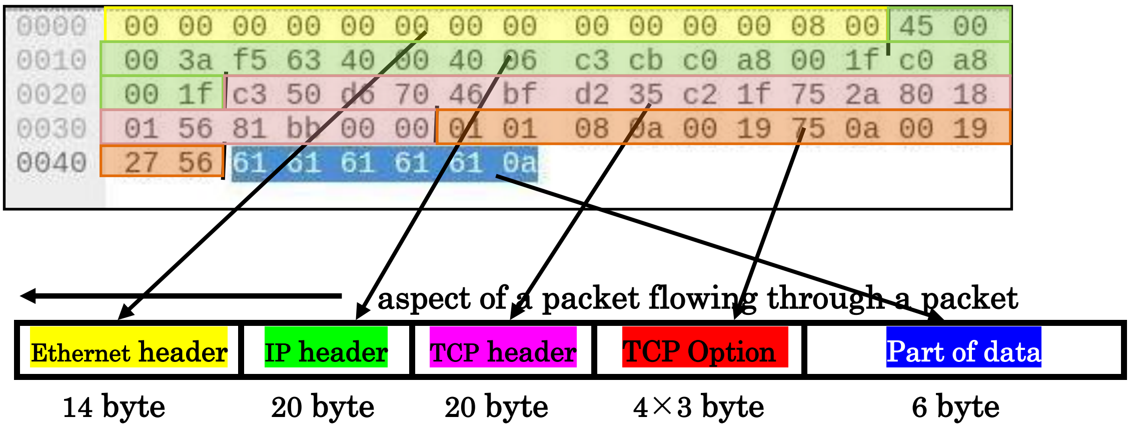
［ｂ］：Contents of a single packet



From top to bottom.

- ･Ethernet header
- ･IPv4 header
- ･TCP header
- ･Order of transmitted data.　The entire packet (Ethernet frame) is 72 bytes.

[c]:A specific hexadecimal and ASCII representation of the contents of a single packet



Tanuki, you can't learn analysis just by looking at what is displayed. I will give you the following exercises to try. Practice is an important part of learning. However, repeating the same thing over and over again to avoid mistakes is a waste of time. So studying for the exam is doing a waste of time. It makes sense if you want to become a bureaucrat who can blame mistakes. However, it is important to practice a few times not to prevent mistakes, but to deepen understanding. If you deny this, you're not learning.

I understand. I've always wanted to try it. But I'd also like to check the answers, so please provide me with the answers.

［Exercise 1] Write the corresponding hexadecimal number from [c] in the following header.

Ethernet header （１４byte）

Destination MAC address（6 byte）	Sender MAC address（6 byte）	Type（2 byte）

IP header（２０byte）

version (4)	header length (4)	DSCP(6)	ECN (2)	packet length (16)

identifier (16)	O(1)	F(1)	M(1)	fragment offset (13)

live time (8)	protocol number (8)	header checksum (16)

starting IP address 192.168.0.31(32 ﾋﾞｯﾄ)

end IP address 192.168.0.31(32 ﾋﾞｯﾄ)

TCP header（２０byte）

starting port number 50000(16 bit)	endpoint port number 54896(16 bit)

sequence number　(32 bit)

acknowledgment number (32 bit)

data offset (4)	reserved bit (3)	control flag (9)	window size (16)

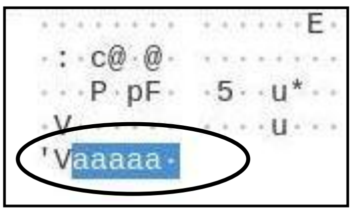
checksum (16 bit)	urgent pointer (16 bit)

TCP option（１２byte）

option

DATA（6 byte）

a	a	a	a	a	nl (LF) new line



［Answers to Exercise 1］

Ethernet header （１４byte）

Destination MAC address (6 byte)	Sender MAC address (6 byte)	Type (2 byte)
00 00 00 00 00 00	00 00 00 00 00 00	08 00

IP header（２０byte）

version (4)	header length (4)	DSCP(6)	ECN (2)	packet length (16)
4	5	0	0	00 3a

identifier (16)	O(1)	F(1)	M(1)	fragment offset (13)
F5 63		4		0　00

live time (8)	protocol number (8)	header checksum (16)
40	06	c3 cb

starting IP address 192.168.0.31(32 ﾋﾞｯﾄ)
C0 a8 00 1f

end IP address 192.168.0.31(32 ﾋﾞｯﾄ)
C0 a8 00 1f

TCP header（２０byte）

starting port number 50000(16 bit)	endpoint port number 54896(16 bit)
C3 50	D6 70

sequence number　(32 bit)
46 bf d2 35

acknowledgment number (32 bit)
C2 1f 75 2a

data offset (4)	reserved bit (3)	control flag (9)	window size (16)
8	0　1	8	01 56

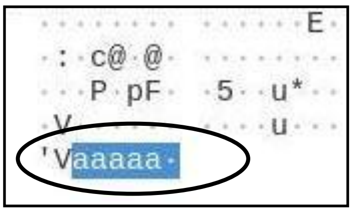
checksum (16 bit)	urgent pointer (16 bit)
81 bb	00 00

TCP option（１２byte）

option
01 01 08 0a 00 19 75 0a 00 19 27 56

DATA（６byte）

a	a	a	a	a	nl (LF) new line
61	61	61	61	61	0a



[Detailed explanations for packet analysis.]

Ethernet Header Terms

Type: If the upper layer is IP, 0x0800. For ARP (Address Resolution Protocol: MAC address query from broadcast address such as ping, dns, nslookup, etc.), 0x0806.

IP Header Terms

Version: 4 for IPv4.

Header length: 4 byte units, so 4 x ? = 20 (bytes) from ? = 5, which is 5.

DSCP、ECN : Indicates the state of packet congestion on the transmission line.

Packet Length: The length (in bytes) of an IP packet. The following calculation formula is used.

Packet length = Length of entire packet (72 bytes) − Ethernet header (14 bytes) = 58 = x003a

Identifier: Increase by 1 for each packet sent out.

Flags: O (unused: 0), F (divisible: 0, not divisible: 1), M (last fragment (not fragmented): 0, fragment in progress: 1)

Fragment Offset: Fragmentation (packet splitting) occurs when a single IP packet exceeds 1500 bytes. Indicates where the fragmented data will be in the original packet; 0x4000 indicates that fragmentation is not possible and no fragmentation has occurred.

Survival Period: Indicates the number of routers an IP packet can pass through. Each time it passes through a router, the number is decreased by 1.

Protocol number: Indicates the upper protocol: 6 for TCP, 17 for UDP, 1 for ICMP, and 4 for IP.

Header checksum: guarantees that the IP header is not corrupted. 1's complement of the IP header.

TCP Header Terms

Sequence number: Order control, restoring the sent TCP packets in the correct order.

Acknowledgement number: Sequence number + data size received

Data offset: TCP header length (5 for 20 bytes). In this case, 32 bytes including the TCP option, so 8 is entered as 32/4.

Reserved bits: 3 bits reserved for future expansion. Currently unused and set to 0.

Control flags: 1-bit flags (9 bits in total) for connection-related control (NS, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN).

Window size: Notifies the receiver of the size of the data to be sent and prepares the receive buffer. In this case, 0x0156 is used to notify the transmission of 342 (decimal) bytes of data.

Checksum: Ensures that TCP packets are not corrupted.

Urgent pointer: Indicates the location and number of bytes of data that must be processed urgently.

※Connectionless UDP does not include the sequence number, acknowledgment number, or control flags required for a connection.

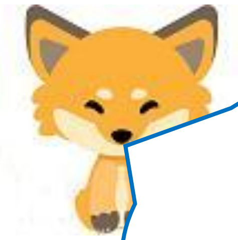
Kitsune, the packets sent and received are flying in and out at a dizzying rate, is there any way to save them to a file and analyze them carefully?

Yes, there is.
If you execute the following command with Root privileges, the file (cap1) will be saved. However, when you open the cap1 file, you must start WireShark and open it as a file for WireShark. At this time, you do not need to have Root privileges to open the file.

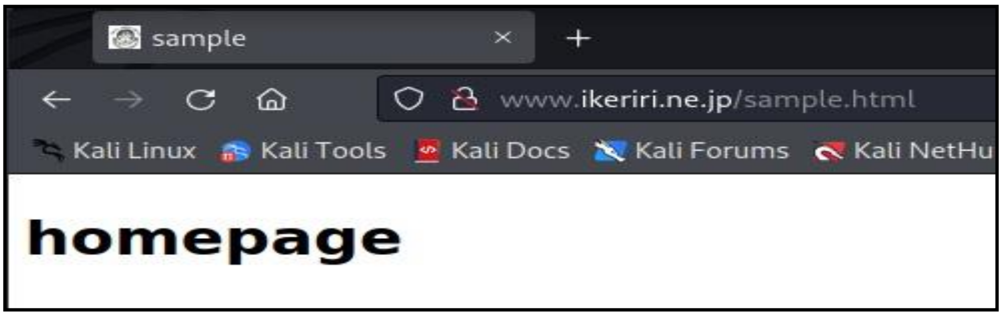
```
]# tshark  -i  eth0  –w  /home/cap1

(Note) The cap1 file created in /home/ does not have access rights, so grant access
       rights as follows.
]# chmod  777  /home/cap1
```

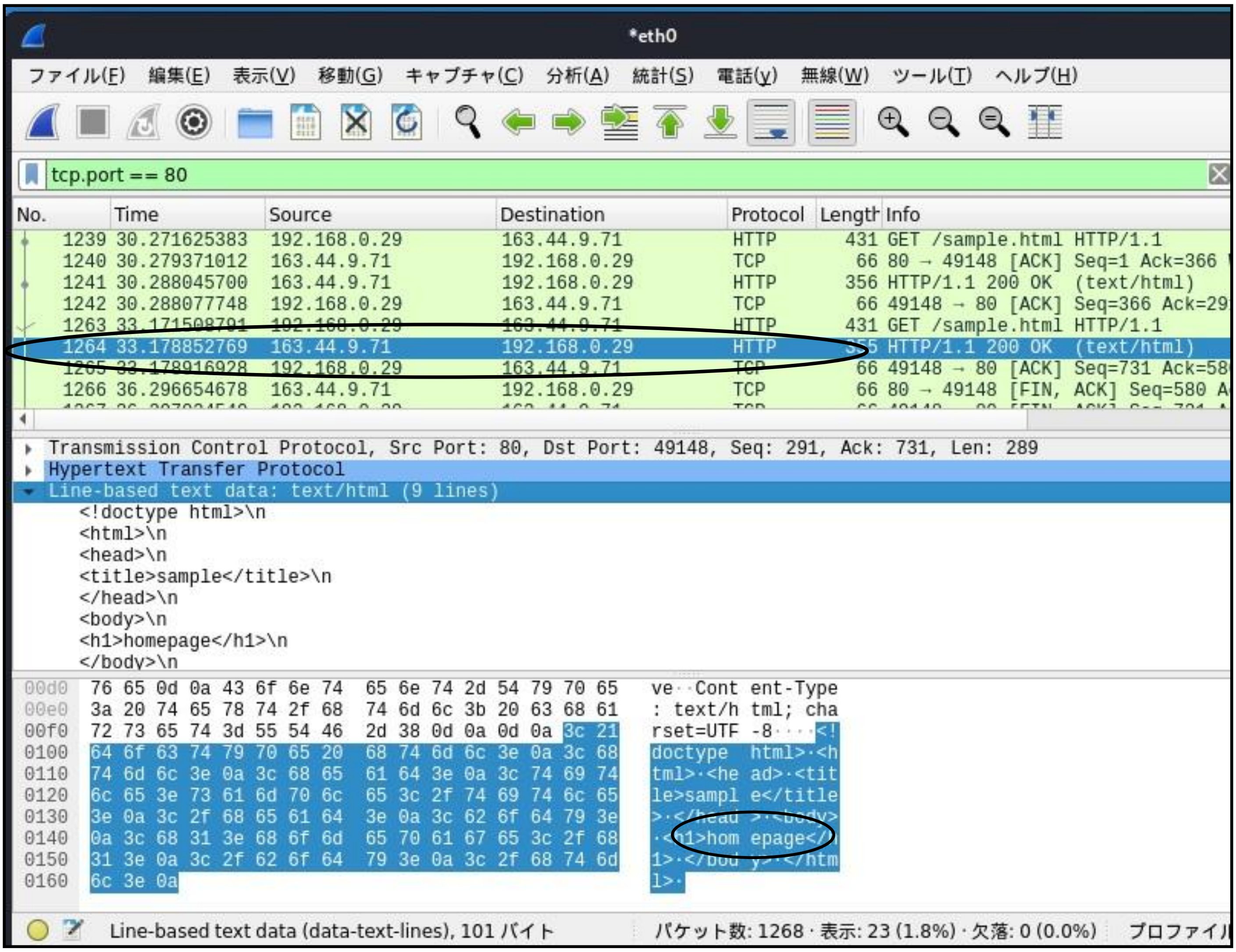
I understand that [Detailed Explanation for Packet Analysis] is an important term, but it's not easy to understand. Do I have to learn this?

You don't have to memorize it, but I'd like you to understand how it works as you do your practice assignments.



Next, I should try to analyze the packets from the web server. There is a site that provides a demo page for analysis, so let's use that. The site is not using OpenSSL, so it is accessible via http, not https, and the HTTP header and data portions are clearly distinguishable. The following is the result of accessing this site and capturing the data with WireShark. The IP address of "www.ikeriri.ne.jp" is 163.44.9.71. The IP address of the "kali Linux" that is capturing the packets is 192.168.0.29.





Tanuki, as [Exercise 2], start wireshark and also start a browser. Access the "http://www.ikeriri.ne.jp/sample.html" site from your PC. Extract the first and last lines of the http header section from (b) of wireshark at that time.If you are asked to look at part (c) and count the number of bytes in the http header, will you be able to practice and answer the question?

[Answers to Exercise 2]

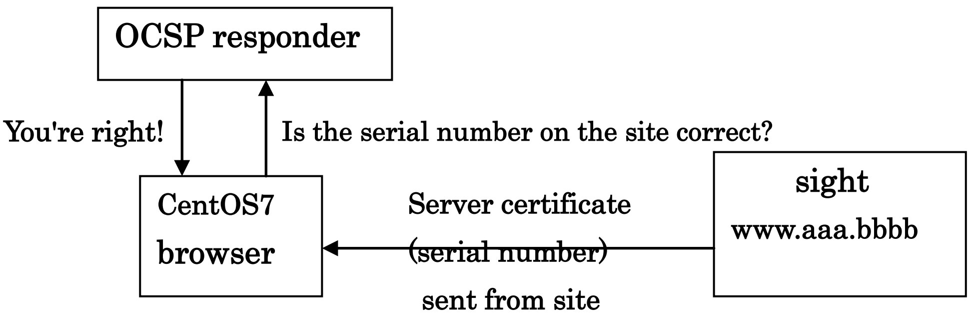
First line : GET / HTTP/1.1
Last line : Connection: Keep-Alive
Bytes: Count them yourself!

Let's continue. Start wireshark and a browser as "Exercise 3". Extract from (b) the line that borders the http header part and the data part when the data is sent from the "http://www.ikeriri.ne.jp/sample.html" site to the PC.
Also, count the number of bytes in the http header section in (c). If you were asked to do this, would you be able to answer the question?

[Answers to Exercise 3]

boundary line : Content-Type: text/html
or blank (¥r¥n: carriage return/line feed)
Bytes: Count them yourself!

Now, when I capture https using WireShark, I see OCSP protocol packets. What is OCSP? It is a packet that queries an external OCSP responder to see if the digital certificate sent by the site you are accessing is correct. It is shown in the next figure.



So [exercise 4].
Launch WireShark, then launch your browser and visit the "https://www.yahoo.co.jp" site.
Answer the encryption name used in the OCSP protocol hash algorithm displayed at that time, the number of bytes of the issuerNameHash (issuer name), the number of bytes of the issuerKeyHash (issuer's public key) and the serial number.

[Answers to Exercise 4] OCSP Protocol

Hash algorithm： ＳＨＡ－１
issuerNameHash(Publisher Name)：20 byte
issuerKeyHash(Public key of the issuer)：20 byte
serial number：16 byte

Next is the UDP protocol.
Capturing the DNS protocol using WireShark shows that the UDP protocol is used.
So [exercise 5].
Looking at the displayed UDP header, can you fill in the following table with the header contents in hexadecimal?

UDP header

Starting port number(2byte)	Endpoint port number(2byte)
Packet length(2byte)	Checksum(2byte)

[Answers to Exercise 5] UDP header

Starting port number （２byte)	Endpoint port number （２byte)
A4 7d	00 35 (53:DNS Server)
Packet length （２byte)	Checksum （２byte)
00 2ｂ (43 byte)	60 e8

[Exercise 6] and it's the last one.
Launch WireShark (specify DNS protocol), then launch your browser and read the following flags from the DNS header that appears when you access the "https://www.yahoo.co.jp" site .
First, how many bytes are in the DNS header?
One thing to note, however.
There are two types of DNS headers, one in the request packet and the other in the response packet. Both have the same format shown in the table below, but the values are different.

[Requests.]

I D (16 bit)									
Expressed in hexadecimal :									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
	binary digits :								

[Response.]

I D (16 bit)									
Expressed in hexadecimal :									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
	binary digits :								

[Answers to Exercise 6] DNS Header

[Requests.]

I D (16 bit)									
Expressed in hexadecimal : 5d 30									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
0	binary : 0000	0	0	1	0	0	0	0	0000

01 00 (hexadecimal)

[Response.]

I D (16 bit)									
Expressed in hexadecimal : 5d 30									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
1	binary : 0000	0	0	1	1	0	0	0	0000

81 80 (hexadecimal)

Finally, an encore to the exercise.
Consider what can be read from the above table from the bit sequence of flags, divided into [request] and [response].

[Answers to Encore for Practice] DNS Header

> [request] : QR=0 indicates a query.Opcode=0 indicates a normal query; RD=1 indicates a full-service resolver.
> [response] : QR=1, response. From Opcode=0, normal query. From RD=1, full service resolver. From RA=1, it is clear that name resolution is possible.

I'll give you a detailed supplementary explanation of DNS headers, see if you need it.

[Supplemental DNS header description.]

The DNS header (application layer) is shown in the table below.

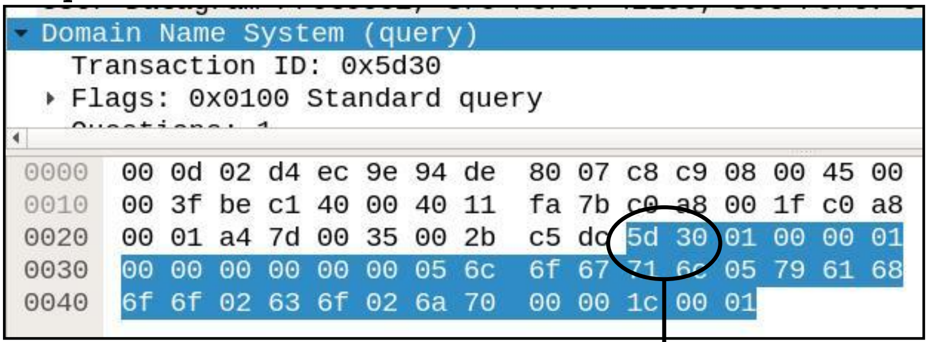
I D (16 bit)									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
QDCOUNT(16 bit)									
ANCOUNT(16 bit)									
NSCOUNT(16 bit)									
ARCOUNT(16 bit)									

ID: Specified at the time of query (inquiry) and copied at the time of response (reply).
QR:Inquiry 0, response 1.
Opcode:Normal query 0, Notify is 4, and Update is 5.
:
RD:Name resolution. Query authoritative DNS servers0 and full-service resolvers (DNS servers that look at their own cache and ask them to tell you if they don't know)1.

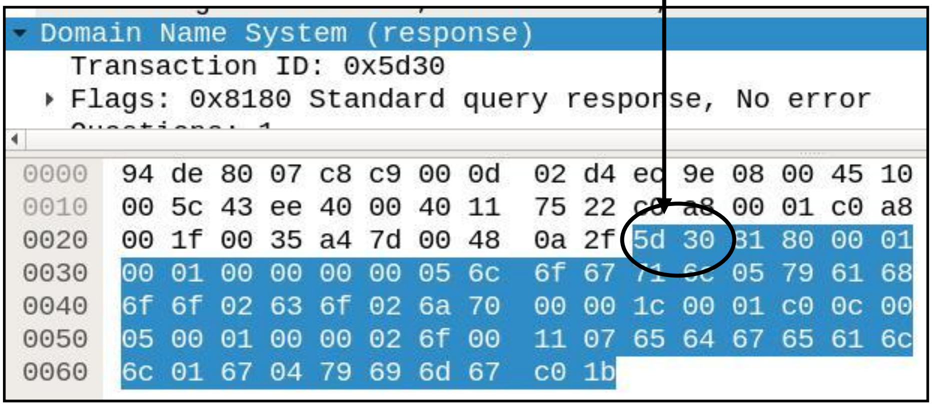
RA : Name resolution is possible 1.
Z:Future Reservations. Always 0.
:

As shown in the figure below, there are two types of DNS headers: packets in the request and packets in the response.Both have the same format shown in the table above, but the values are different.

[request]



[Response.] ID is copied.



A DNS packet is formed by a DNS header + data (Question section, Answer section, Authority section, and Additional section).
The data portion also includes a variable-length portion by domain name.

Kitsune, I'm exhausted from all the exercises and explanations! Let me take a break.

Yes, it is tiring, isn't it? I know how it feels because it was hard for me to understand it too.
But you know, it is the basic knowledge to hack and to defend against hacking. If you don't get over this, it's a dream come true to get advanced knowledge. That's the end of the story about the contradictions and shields of security. What shall we talk about in Episode 28?

Translated at DeepL