第２７話（セキュリティⅥ：矛）　パケット解析Ⅱ

ＷｉｒｅＳｈａｒｋの利用

タヌキ、次はＷｉｒｅＳｈａｒｋを使ったパケット解析だ。解析に使用する題材は、第６話と第７話で作成したチャットプログラムだ。サーバ設定のチャットとクライアント設定のチャット間のコミュニケーション時のパケットを Wireshark で解析することにする。ＡＳＣⅡコードも必要になると思うので、再度提示したままにする。

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

ＩＴ用語辞典より

タヌキは、tcpdump で勉強したから、通信回線の中をデータが流れる時、パケットという単位に区切られて送信されていることを知っているよな。ここで復習だ。
１個のパケットは区切られたデータに上位のアプリケーション層（例えば、HTTP）のヘッダ、トランスポート層（例えば、TCP）のヘッダ、ネットワーク層（例えば、IP）のヘッダ、個別ネットワーク層（例えば、Ethernet）のヘッダからできているよな。
図示すれば、以下のようになる。

パケットが流れる向き

Ethernet ヘッダ	IP ヘッダ	TCP ヘッダ	データの一部	FCS
14 バイト	20 バイト	20 バイト （UDP：8 バイト）		4 バイト トレイラ

※FCS（トレイラ）：パケットが転送中に壊れなかったどうかチェックする。

PC（A）から PC（B）にパケットが流れる時、そのパケットを取り込むことをパケットキャプチャと言ったよな。またキャプチャしたパケットの中を見ることをパケットモニタリングという。実際に流れているパケットを見て解析することは、書籍で覚えるよりも身に付くことは言うまでもないだろう。ただ、パケットモニタリングするための操作方法と見方を学ばなければ、猫に小判だよな！
前にも言ったけれども、「kali Linux」には WireShark がインストールされているので、直ぐに使えるよ。

「kali Linux」以外で WireShark がインストールされていない場合はどうするのだ。例えば、オイラが構築した CentOS 7 で使いたい場合には？

CentOS7 にインストールすることはできるが、結構大変な作業になるのだ。「kali Linux」をインストールした方が早いと思うが、一応 CentOS7 にインストールする手順を書いておく。

[Wiresshark のインストール]

① 先ずは準備

$ su – root　　# root になる

mkdir tool　　# root に　保存用のディレクトリ tool を作成

② 次に Openssl が必要

CentOS に Openssl がインストールされているかどうか確認する。

```
# openssl version
```

インストールされていた場合、念のために以下の作業を行なう。

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

(新たに Openssl をインストールする場合)

依存関係のインストール

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

Openssl-1.1.1 ソースのダウンロード

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

Openssl-1.1.1 のインストール

```
# 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
```

インストール確認

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

動作確認

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

Root のカレントディレクトリに戻る。

```
# cd ~
```

③既に Openssl がインストールされていても新規にインストールした場合でも、共通して行なわなければならない作業

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

④　python3 が必要

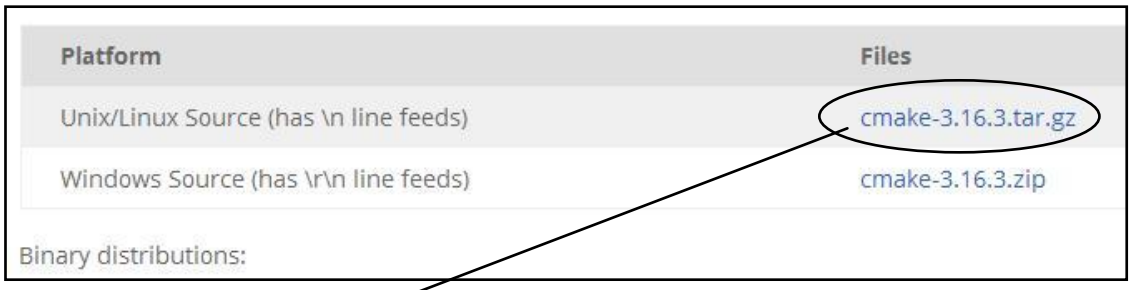
```
# yum install python3
```

⑤　cmake が必要

ダウンロードサイト

https://cmake.org/download/

サイトの表示は以下の通り



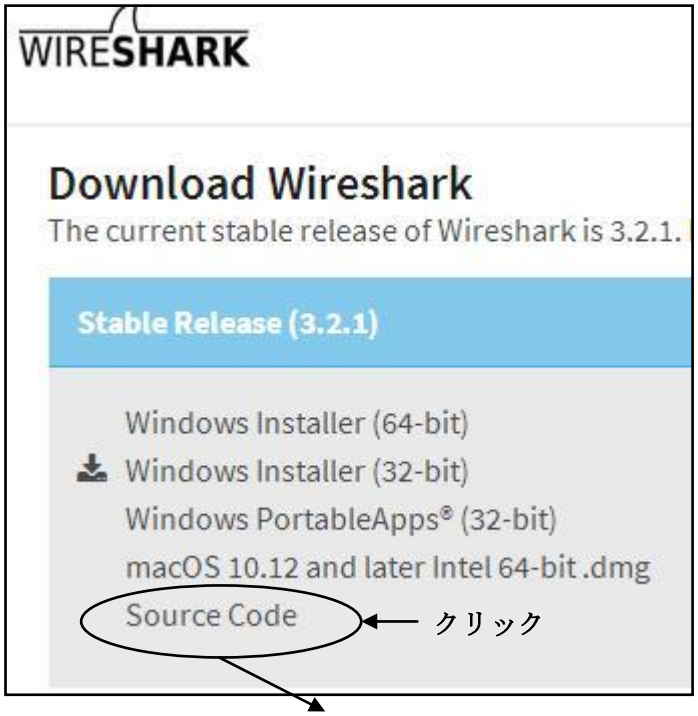
/root/tool/cmake-3.16.3.tar.gz となるように保存

```
# cd tool
解凍
# tar xvfz ./cmake-3.15.3.tar.gz
解凍したディレクトリに移動
# cd ./cmake-3.15.3
必要なパッケージのインストール
# yum install -y gcc gcc-c++
bootstrap で makefile を作成
# ./bootstrap
make & make install
# make
# make install
確認
# cmake --version
```

WireShark のインストールに cmake はどうして必要なのだ？

WireShark をソースファイルで取得するので、ビルド（コンパイルを含む）しなければならない。それに必要なのが cmake なのだ。
いよいよ、WireShark のダウンロード&インストールだ。

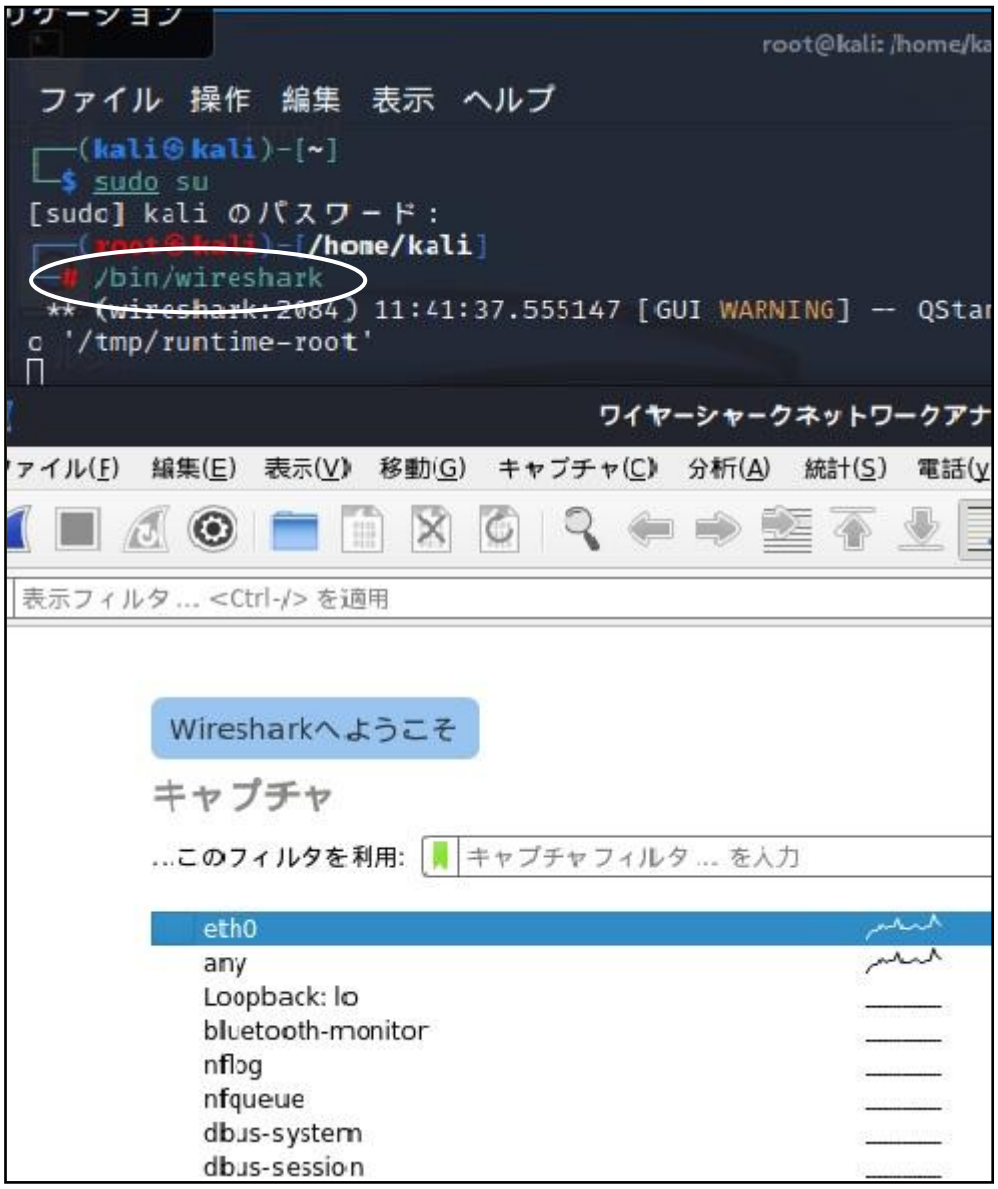
⑥ WireShark のインストール
ダウンロードサイト
https://www.wireshark.org/download.html



/root/tool/wireshark-3.2.1.tar.xz となるように保存

cd tool
解凍
```
# tar xJfv ./wireshark-3.0.5.tar.xz
```
ビルド用のディレクトリを作成し、入る
```
# mkdir build
# cd build
```
必要なパッケージのインストール
```
# 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
```
cmake の実行
```
# cmake ../wireshark-3.0.5
```
make & install
```
# make
# make install
```
確認
```
# tshark --version
```
TShark (Wireshark) 3.0.5 と表示されれば完了です。

WireShark ツールを使えるようになったので、操作方法も兼ねて、Web サーバからのパケット解析でもやってみるか。ただ、以下のように管理者権限（root）で WireShark を起動しないと、解析に必要な完全パケットを取得することができないからな。WireShark のアイコンをクリックしても管理者権限にはならないよ！



読み取れるぞ！なるほど、端末で Root になってから WireShark を起動するのか。eth0 は NIC のデバイス名か。それを指定するのだな。すでにパケットが届いているようだが。でも、操作に馴れないと難しそうだな。

キツネ、パケット解析に必要な操作方法は説明するが、WireShark の種々の操作については、ネットや本を参考に自分で勉強してね。前から言っているように、ここでは、ツールやアプリの操作方法を学ぶことを目的にはしていないからね。

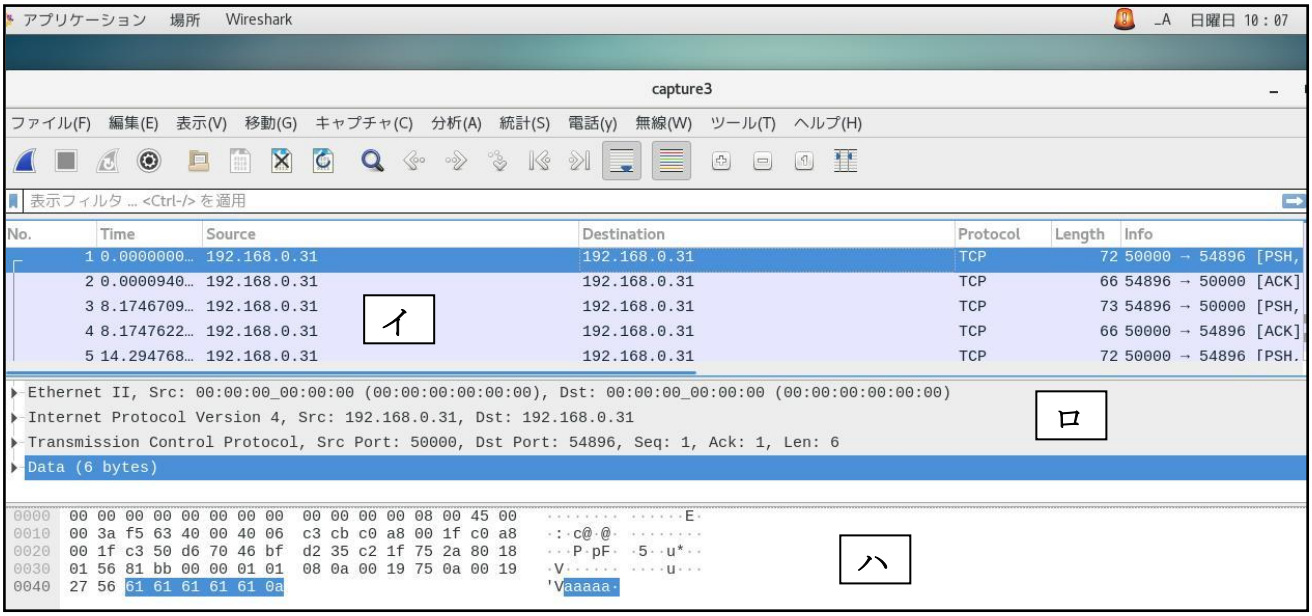
了解、本を読める能力があれば操作方法は理解できるからね。

WireShark を使ってパケットの解析をする為には、事前に送受信の内容がわかっている、わかり易いテスト用のデモモデルを使うのが良い。それで、第６話と第７話で作成したチャットプログラムを使うことにする。サーバ用チャットとクライアント用チャットを使えば、送信内容が事前にわかっているし、相互通信なので最適だ。
では、先ず、チャットのプログラムを使ってパケットの解析だ。チャットのプログラムは１台のＰＣで実行している。

［パケット解析用チャットプログラムの実行状態］

WireShark によるパケット解析は、以下のようになる。
表示されるイ、ロ、ハの部分の意味を良く理解することが大切だ。それができなければ、ハッキングなんて夢の又夢だ。頑張ってみてくれたまえ。

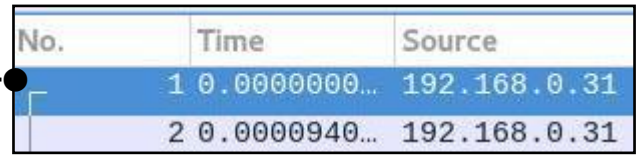
[Wireshark によるパケット解析の基本]



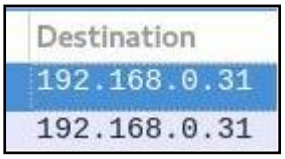
上記を［イ］、［ロ］、［ハ］の３部分に分けて説明する。

［イ］：１行が１パケットを表す。

パケットの送信順：経過時間（最初は０）：送信 IP アドレス　受信側 IP アドレス
（サーバプロセス）　（クライアントプロセス）



No.	Time	Source
1	0.0000000…	192.168.0.31
2	0.0000940…	192.168.0.31



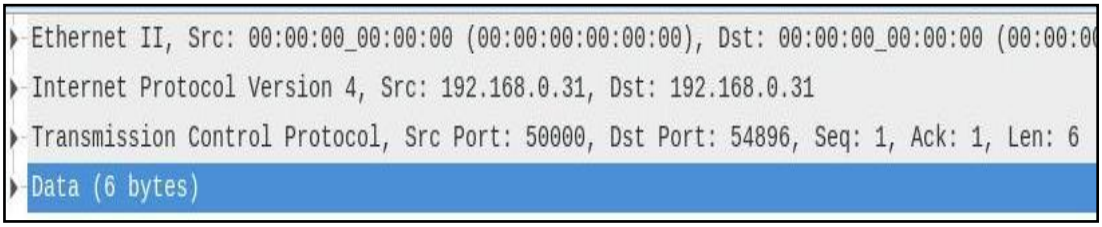
Destination
192.168.0.31
192.168.0.31

使用プロトコル：フレームの長さ（バイト）：送信ポート（50000）→受信ポート（54896）

Protocol	Length	Info
TCP	72	50000 → 54896 [PSH,
TCP	66	54896 → 50000 [ACK]

通信制御フラグ群
PSH（プッシュ）：送信準備
ACK：応答確認
：

［ロ］：１個のパケットの内容



上から順に

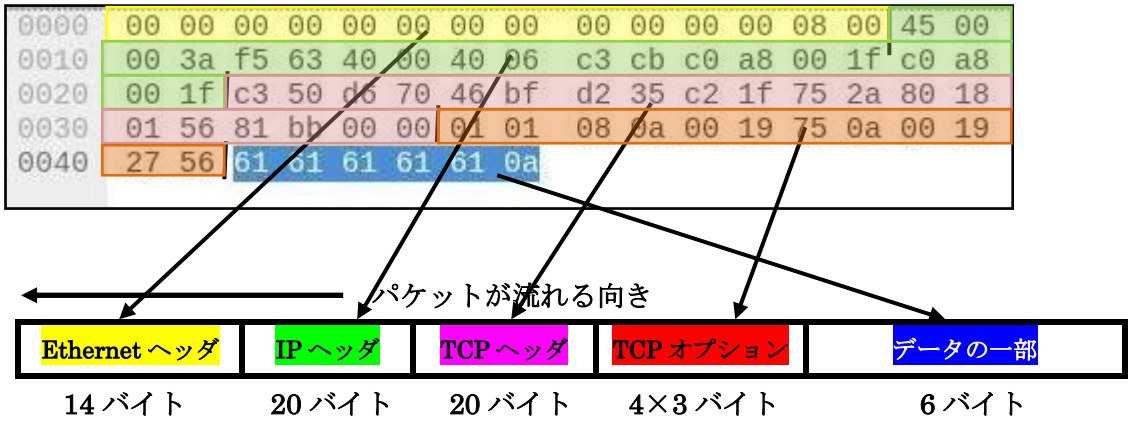
・Ethernet ヘッダ

・IP v 4 ヘッダ

・TCP ヘッダ

・送信データ　の順。　全体が 1 パケット（ｲｰｻﾈｯﾄﾌﾚｰﾑ）72 バイトである。

［ハ］：１個のパケットの内容を具体的に１６進と ASCII で表したもの



タヌキ、表示された物をただ眺めているだけでは解析の勉強にはならないぞ。次のような練習課題を与えるからやってみなさい。勉強する上で練習は大切な部分だ。ただ、ミスを防ぐ為に同じ事を何度も繰り返すのは、時間の無駄だ。だから受験勉強は時間の無駄を行っているのだ。ミスを咎められる官僚になるなら意味があるけどな。ただ、ミスを防ぐ為では無く、理解を深める為に数回行う練習は大切な作業だ。これを否定したら勉強にならないぞ。

わかった。オイラもやってみたいと思っていたんだ。ただ、答え合わせもしたいので、解答も提示してくれな。

［練習課題 1］以下のヘッダに［ハ］から該当する１６進数を書き込みなさい。

Ethernet ヘッダ（１４バイト）

送信先 MAC アドレス（6ﾊﾞｲﾄ）	送信元 MAC アドレス(6ﾊﾞｲﾄ)	タイプ (2ﾊﾞｲﾄ)

IP ヘッダ（２０バイト）

ﾊﾞｰｼﾞｮﾝ(4)	ﾍｯﾀﾞ長(4)	DSCP(6)	ECN (2)	パケット長(16)			
識別子(16)				O(1)	F(1)	M(1)	ﾌﾗｸﾞﾒﾝﾄｵﾌｾｯﾄ(13)
生存時間(8)		ﾌﾟﾛﾄｺﾙ番号(8)		ﾍｯﾀﾞﾁｪｯｸｻﾑ(16)			
始点 IP アドレス 192.168.0.31(32ﾋﾞｯﾄ)							
終点 IP アドレス 192.168.0.31(32ﾋﾞｯﾄ)							

TCP ヘッダ（２０バイト）

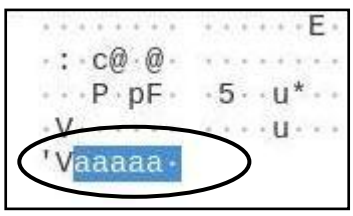
始点ポート番号 50000(16ﾋﾞｯﾄ)		終点ポート番号 54896(16ﾋﾞｯﾄ)	
シーケンス番号(32ﾋﾞｯﾄ)			
確認応答番号(32ﾋﾞｯﾄ)			
ﾃﾞｰﾀｵﾌｾｯﾄ(4)	予約ﾋﾞｯﾄ(3)	ｺﾝﾄﾛｰﾙﾌﾗｸﾞ(9)	ウインドウサイズ(16)
チェックサム(16ﾋﾞｯﾄ)		緊急ポインタ(16ﾋﾞｯﾄ)	

TCP オプション（１２バイト）

オプション

DATA（６バイト）

a	a	A	a	a	nl (LF)改行



［練習課題１の解答］

Ethernet ヘッダ（１４バイト）

送信先 MAC アドレス（6ﾊﾞｲﾄ）	送信元 MAC アドレス(6ﾊﾞｲﾄ)	タイプ (2ﾊﾞｲﾄ)
00 00 00 00 00 00	00 00 00 00 00 00	08 00

IP ヘッダ（２０バイト）

ﾊﾞｰｼﾞｮﾝ(4)	ﾍｯﾀﾞ長(4)	DSCP(6)	ECN (2)	パケット長(16)			
4	5	0	0	00 3a			
識別子(16)				O(1)	F(1)	M(1)	ﾌﾗｸﾞﾒﾝﾄｵﾌｾｯﾄ(13)
F5 63					4		0 00
生存時間(8)		ﾌﾟﾛﾄｺﾙ番号(8)		ﾍｯﾀﾞﾁｪｯｸｻﾑ(16)			
40		06		c3 cb			
始点 IP アドレス 192.168.0.31(32 ﾋﾞｯﾄ)							
C0 a8 00 1f							
終点 IP アドレス 192.168.0.31(32 ﾋﾞｯﾄ)							
C0 a8 00 1f							

TCP ヘッダ（２０バイト）

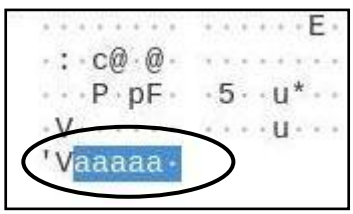
始点ポート番号 50000(16 ﾋﾞｯﾄ)			終点ポート番号 54896(16 ﾋﾞｯﾄ)
C3 50			D6 70
シーケンス番号(32 ﾋﾞｯﾄ)			
46 bf d2 35			
確認応答番号(32 ﾋﾞｯﾄ)			
C2 1f 75 2a			
ﾃﾞｰﾀｵﾌｾｯﾄ(4)	予約ﾋﾞｯﾄ(3)	ｺﾝﾄﾛｰﾙﾌﾗｸﾞ(9)	ウインドウサイズ(16)
8	0 1	8	01 56
チェックサム(16 ﾋﾞｯﾄ)			緊急ポインタ(16 ﾋﾞｯﾄ)
81 bb			00 00

TCP オプション（１２バイト）

オプション
01 01 08 0a 00 19 75 0a 00 19 27 56

DATA（６バイト）

A	a	a	a	a	nl (LF)改行
61	61	61	61	61	0a



［パケット解析の為の詳細な解説］

Ethernet ヘッダの用語

タイプ: 上位層がＩＰの場合は、0x0800　ARP (Adress Resolution Protocol : ping、dns、nslookup などブロードキャストアドレスからの MAC アドレスの問い合わせ）の場合は、0x0806

IP ヘッダの用語

バージョン：IP v 4 の場合は、４

ヘッダ長：４バイト単位なので４×？＝２０（バイト）より？＝５で５となる。

DSCP、ECN：伝送路のパケットの混雑状態を表す。

パケット長：IP パケットの長さ（バイト数）です。次の計算式になります。

ﾊﾟｹｯﾄ長　＝　ﾊﾟｹｯﾄ全体の長さ（72ﾊﾞｲﾄ）－Ethernet ﾍｯﾀﾞ（14ﾊﾞｲﾄ）

＝　58　＝　0x003a

識別子：送出されるパケット毎に１増やす。

フラグ：O（未使用：0)、F（分割可：0、分割不可：1)、M（最後のフラグメント（フラグメントしていない）：0、途中のフラグメント：1）

フラグメントオフセット: 1 個の IP パケットが 1500 バイトを越えるとフラグメンテーション(パケット分割）が起こります。分割されたデータが元のパケットのどの位置になるかを示したもの。0x4000 は分割不可でフラグメンテーションしていないことを表す。

生存期間：IP パケットが通過できるルータの数を示す。ルータを通過する度に１づつ減る。

プロトコル番号：上位のプロトコルを表す。TCP は 6、UDP は 17、ICMP は 1、IP は 4

ヘッダチェックサム：IP ヘッダが壊れていないことを保証する。IP ヘッダの１の補数。

TCP ヘッダの用語

シーケンス番号：順序制御、送られたＴＣＰパケットを正しい順番に復元する。

確認応答番号：シーケンス番号＋受信したデータサイズ

データオフセット：TCP ヘッダ長（20 バイトの場合 5)。今回は、TCP オプションを含み 32 バイトなので 32/4 で 8 が入る。

予約ビット：将来の拡張の為に用意されている 3 ビット。現在は未使用で 0 がセット。

コントロールフラグ：コネクションに関する制御をする為の１ビット（合計９ビット）のフラグ（NS,CWR,ECE,URG,ACK,PSH,RST,SYN,FIN)。

ウインドウサイズ：送信するデータサイズを受信相手に通知し、受信バッファの準備をさせる。今回は、0 x 0156 なので 342（10 進数）バイトのデータの送信を通知している。

チェックサム：TCP パケットが壊れていないこと保証します。

緊急ポインタ：緊急に処理しなければならないデータの場所とバイト数を表す。

※ コネクションレス型の UDP には、コネクションに必要なシーケンス番号、確認応答番号、コントロールフラグは含まれていません。

キツネ、送受信のパケットが、目まぐるしく飛び交うのだが、ファイルに保存し、じっくり解析する方法はないのか。

あるよ。Root 権限で以下のようにコマンドを実行するとファイル（cap1）に保存されるよ。ただし、cap1 ファイルを開く時には、WireShark を起動し、WireShark 用のファイルとして開かなければダメだよ。この時は、Root 権限でなくても開けるよ。

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

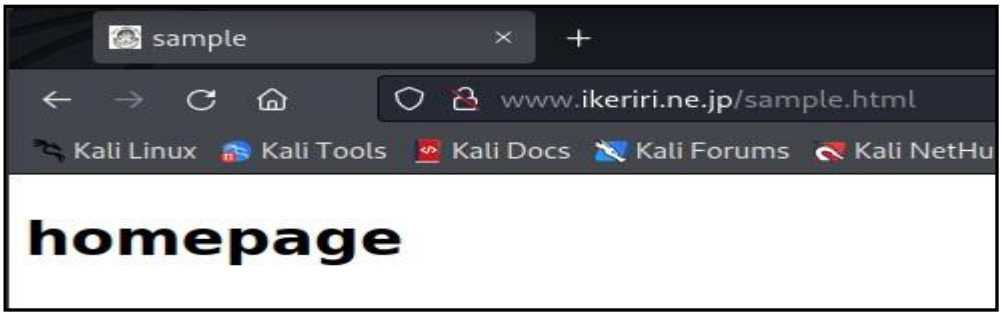
(注)　/home/に作成された cap1 ファイルはアクセス権が無いので、以下のようにアクセス権を付与する。

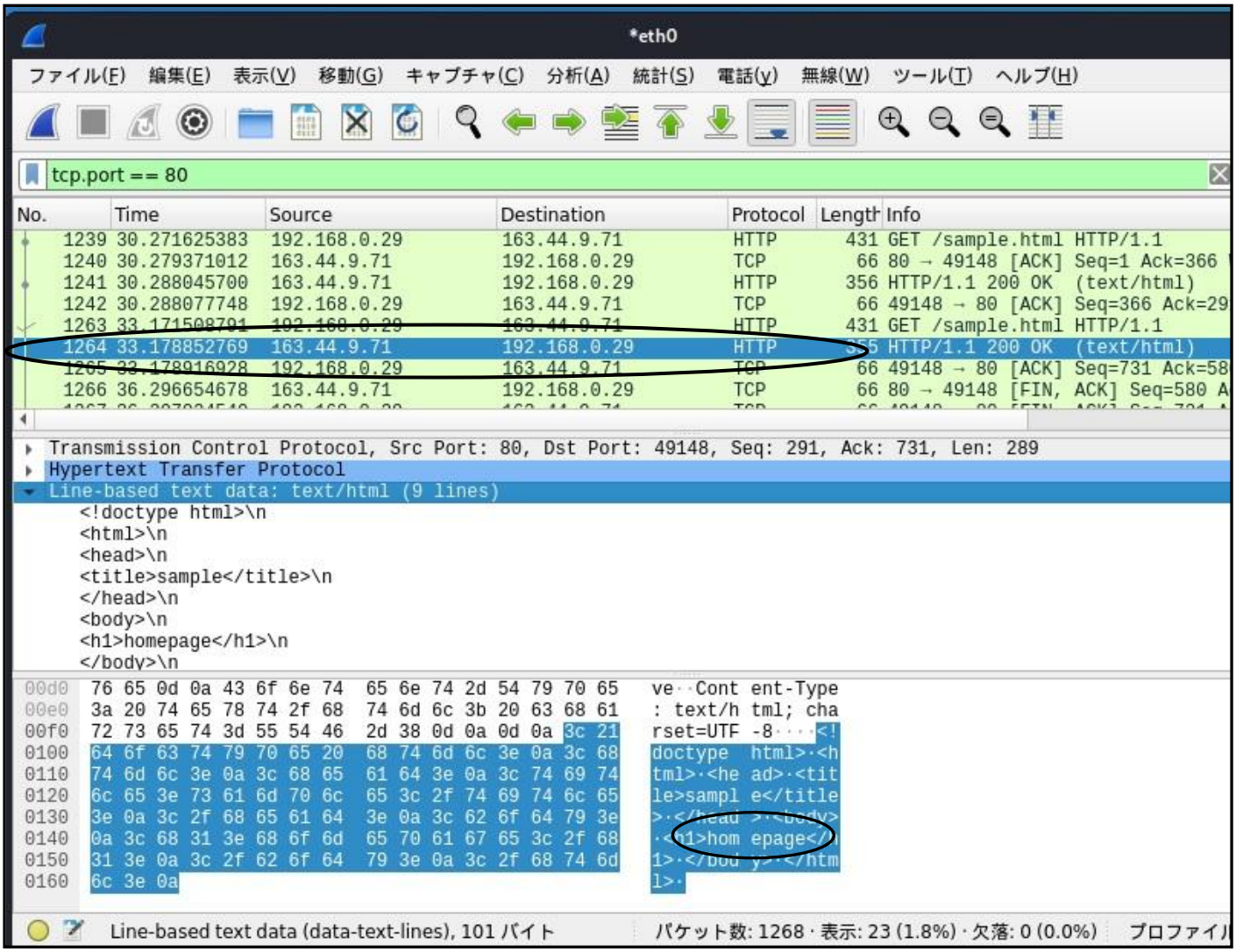
```
]# chmod   777   /home/cap1
```

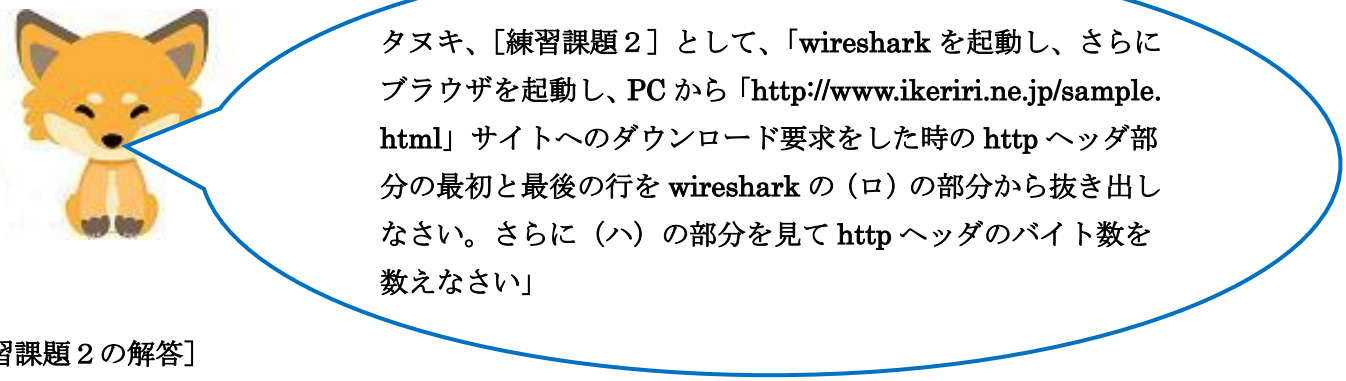
［パケット解析の為の詳細な解説］は、重要な用語なのは理解できるが、なかなか難しいな。これを覚えなければならないのか。

覚える必要は無いけれど、練習課題をしながら仕組みを理解してほしいな。

次は、Web サーバからのパケットの解析をやってみるか。
解析用にデモ用のページを提供してくれているサイトがあるので、それを使ってみよう。OpenSSL を使っていないので、https で無く、http でアクセス可能で HTTP ヘッダ部分とデータ部分が明確に区別できるのだ。該当のサイトは「http://www.ikeriri.ne.jp/sample.html」だ。ここにアクセスし、WireShark でキャプチャした結果が以下の通りだ。www.ikeriri.ne.jp の IP アドレスが１６３．４４．９．７１だ。そのパケットをキャプチャしている「kali Linux」の IP アドレスが１９２．１６８．０．２９だ。

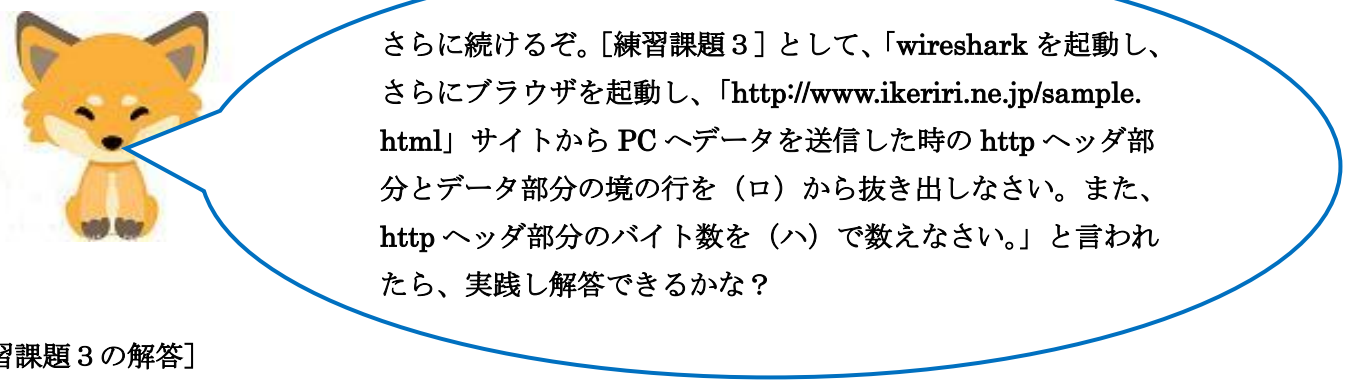






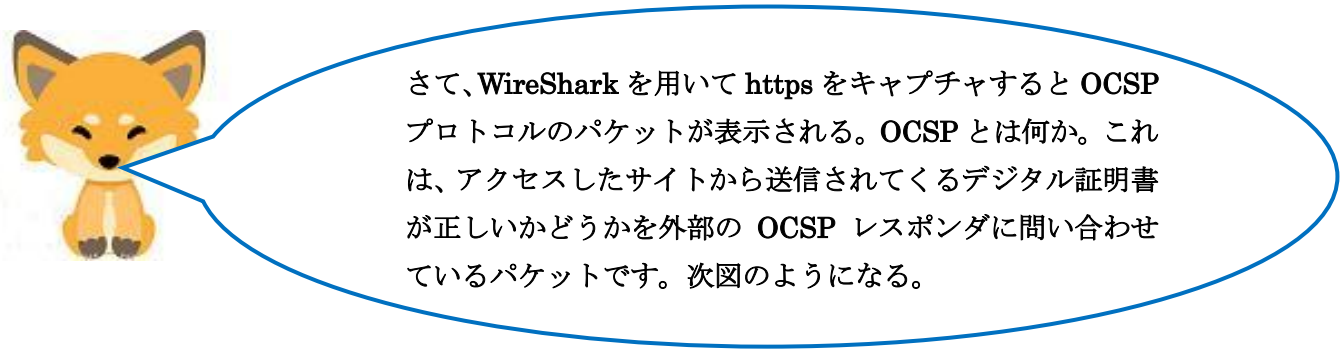
[練習課題２の解答]

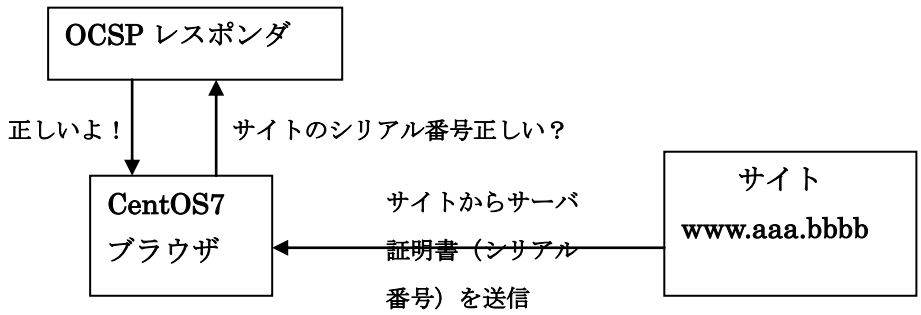
最初の行：GET / HTTP/1.1
最後の行：Connection: Keep-Alive
バイト数：自分で数えましょう



[練習課題３の解答]

境の行：Content-Type: text/html
あるいは 空白（¥r¥n：復帰・改行）
バイト数：自分で数えましょう





それで［練習課題４］だ。WireShark を起動し、さらにブラウザを起動し、https://www.yahoo.co.jp サイトにアクセスした時に表示される OCSP プロトコルからハッシュアルゴリズムに使われている暗号化名、issuerNameHash(発行者名)のバイト数、issuerKeyHash(発行者の公開鍵)のバイト数、シリアル番号のバイト数を答えなさい。

［練習課題４の解答］OCSP プロトコル

ハッシュアルゴリズム：ＳＨＡ－１ issuerNameHash(発行者名)：20 バイト issuerKeyHash(発行者の公開鍵)：20 バイト シリアル番号：16 バイト

次は UDP プロトコルだ。
WireShark を用いてＤＮＳプロトコルをキャプチャするとUDP プロトコルが使われていることがわかる。それで［練習課題５］だ。表示された UDP ヘッダを見て、以下の表にヘッダの内容を１６進数で記入することができるかな。

UDP ヘッダ

始点ポート番号（２バイト）	終点ポート番号（２バイト）
パケット長（２バイト）	チェックサム（２バイト）

［練習課題５の解答］UDP ヘッダ

始点ポート番号（２バイト）	終点ポート番号（２バイト）
A4 7d	00 35　(53:DNS サーバ)
パケット長（２バイト）	チェックサム（２バイト）
00　2b（43 バイト）	60　e8

［練習課題６］で最後だ。
WireShark（DNS プロトコルを指定）を起動し、さらにブラウザを起動し、https://www.yahoo.co.jp サイトにアクセスした時に表示される DNS ヘッダから以下のフラグの情報を読み取りなさい。最初に DNS ヘッダのバイト数は何バイトですか？ただ、１つ注意しておくことがある。DNS ヘッダは、リクエスト時のパケットとレスポンス時のパケットの２種類ある。両者とも下記の表で示したフォーマットは同じだが、数値がことなる。

［リクエスト］

ＩＤ（16 ビット）									
16 進数で表記：									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
	2 進数：								

［レスポンス］

ＩＤ（16 ビット）									
16 進数で表記：									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
	2 進数：								

［練習課題６の解答］DNS ヘッダ

［リクエスト］

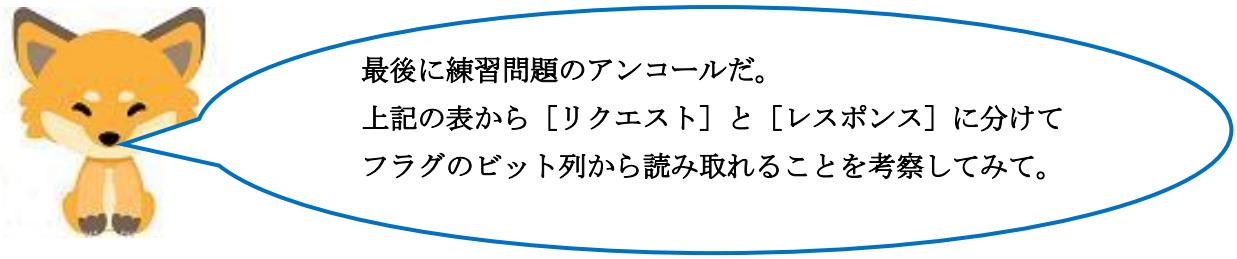
ＩＤ（16 ビット）									
16 進数で表記：				5ｄ　30					
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
0	2 進数：0000	0	0	1	0	0	0	0	0000

01　00　（16 進数）

［レスポンス］

ＩＤ（16 ビット）									
16 進数で表記：				5ｄ　30					
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
1	2 進数：0000	0	0	1	1	0	0	0	0000

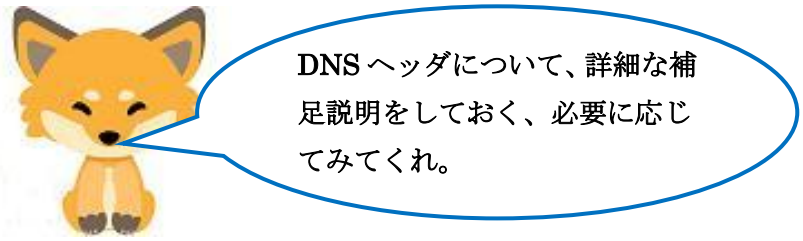
81　80　（16 進数）



［練習課題のアンコールの解答］DNS ヘッダ

［リクエスト］: QR=0 より、問い合わせ。Opcode=0 より、通常の問い合わせ。RD=1 より、フルサービスリゾルバであることがわかる。

［レスポンス］: QR=1 より、応答。Opcode=0 より、通常の問い合わせ。RD=1 より、フルサービスリゾルバ。RA=1 より名前解決が可能であることがわかる。



［DNS ヘッダの補足説明］

ＤＮＳヘッダ（アプリケーション層）は以下の表のようになっている。

ＩＤ（16 ビット）									
QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	Z(1)	AD(1)	CD(1)	RCODE(4)
QDCOUNT(16 ビット)									
ANCOUNT(16 ビット)									
NSCOUNT(16 ビット)									
ARCOUNT(16 ビット)									

ID：クエリ（問い合わせ）時に指定し、レスポンス（応答）時にコピーされる。

QR：問い合わせ 0、応答が 1。

Opcode：通常の問い合わせ 0、Notify は 4、Update は 5。

：

RD：名前解決。権威 DNS サーバへの問い合わせ 0、フルサービスリゾルバ（自分のキャッシュを見て、わからなければ教えてもらいに行く DNS サーバ）１。

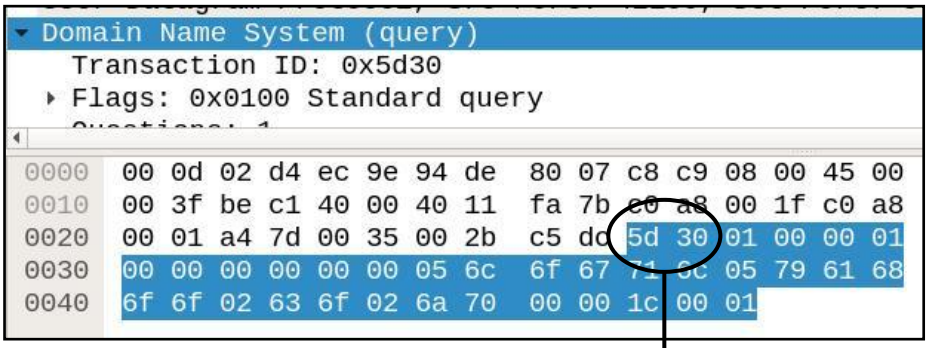
RA：名前解決が可能は１。

Z：将来の予約。常に 0。

：

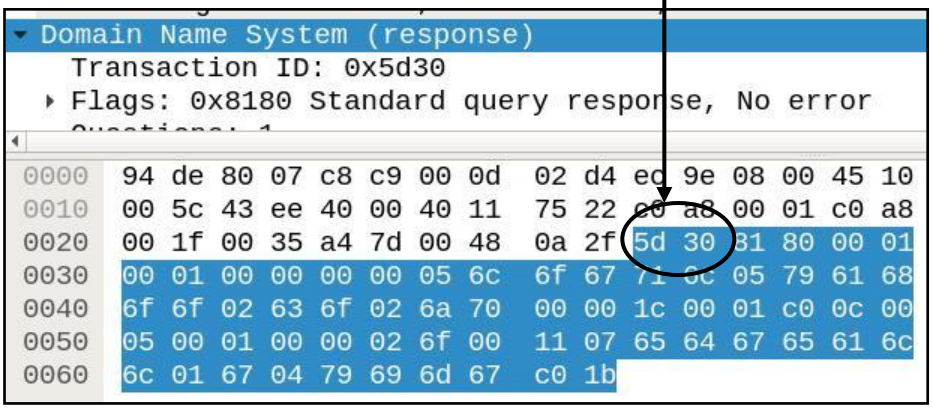
以下の図のように DNS ヘッダは、リクエスト時のパケットとレスポンス時のパケットの２種類ある。両者とも上記の表で示したフォーマットは同じですが、数値がことなる。

［リクエスト］



［レスポンス］

ID コピーされる



DNS パケットは、DNS ヘッダ＋データ（Question セクション、Answer セクション、Authority セクション、Additional セクション）で形成されている。また、データの部分はドメイン名による可変長の部分も含んでいる。

キツネ、多くの練習課題と説明があってオイラ疲れちゃったよ！
少し、休ませてくれ。

確かに、疲れるよな。オイラも理解するのが大変だったので気持ちがわかる。
でもなあ、ハッキングしたり、ハッキングから防御する為の基本的な知識なのだ。これを乗り越えなければ高度な知識を得るのは夢のまた夢だ。セキュリティの矛と盾の話はこれで終わりだ。第２８話は何にしようかな