Network Traffic Analysis

Learning outcome and reading tasks

1. Understand data encapsulation: Stephen Northcutt, Judy Novak. Network Intrusion Detection, section “The TCP/IP Internet model”, page 13 and section “Encapsulation revisited”, page 16-18
2. Dissect a packet: Stephen Northcutt, Judy Novak. Network Intrusion Detection, Third Edition. Sams Publishing 2002 (also available through Metalib, Safari Books Online), section “Dissecting the Whole Packet”, page 140-143
3. Use tcpdump filters: Stephen Northcutt, Judy Novak. Network Intrusion Detection, chapter 2, section “tcpdump output”, page 32-34; http://www.cs.ucr.edu/~marios/ethereal-tcpdump.pdf
4. Craft a TCP packet using hping3
5. Understand IP header fields, TCP header fields, UDP header fields, ICMP header files: read “Help sheet” and “SANS_TCPIP_Cheatsheet_2024” on DLE

Content

- Dissecting a network datagram
- Filtering network traffic
- Examination of Datagram Fields

Packet dissection

IDS – HEX

Intrusion detection system (IDS) is not Plug and Play

- **Human analysis** is necessary to validate intrusion detection systems (IDS) alerts
  - Distinguish false alerts from real attacks
  - IDS/IPS signatures, protocol decodes, traffic analysis far from perfect
- IDS is not Plug and Play. Requires tuning, customisation to improve detection/false alert rates
  - "Crying wolf: False alarms hide attacks"

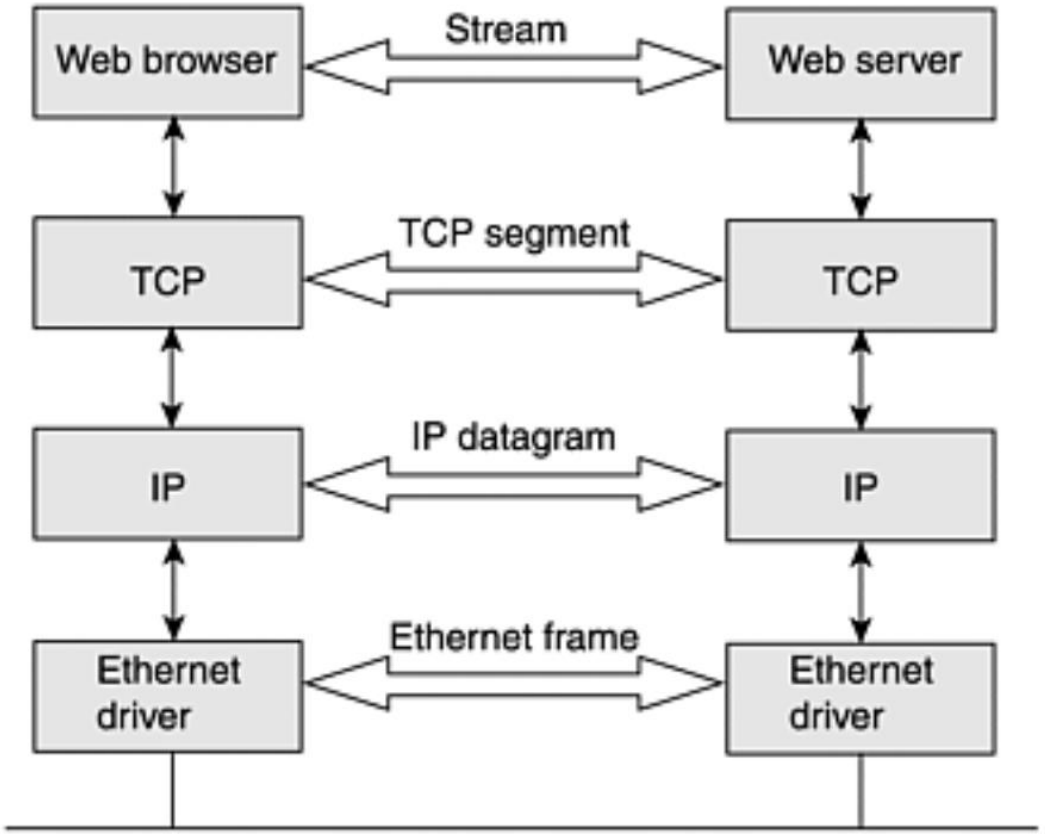
TCP/IP Internet model

APPLICATION LAYER:
Handles implementation of user applications

TRANSPORT LAYER:
Manages end-to-end communications between hosts

NETWORK LAYER:
Gets data from source to destination

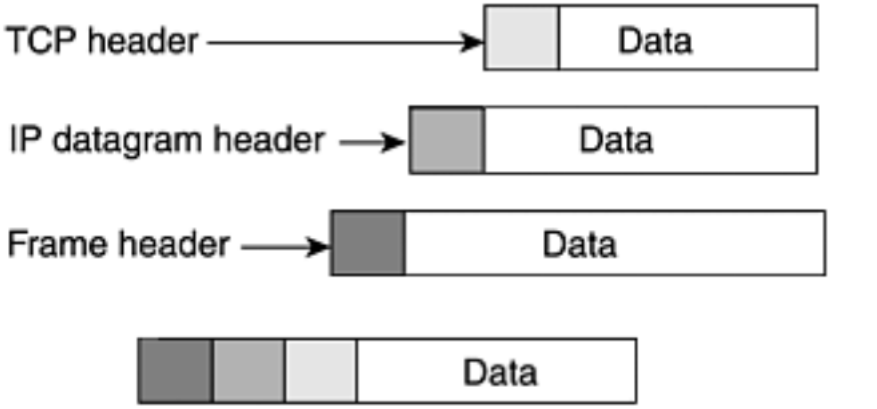
LINK LAYER:
Manages data transfer to and from physical medium



Stephen Northcutt, Network intrusion detection book, figure 1.1, page 12

Data encapsulation

One layer's header is another layer's data

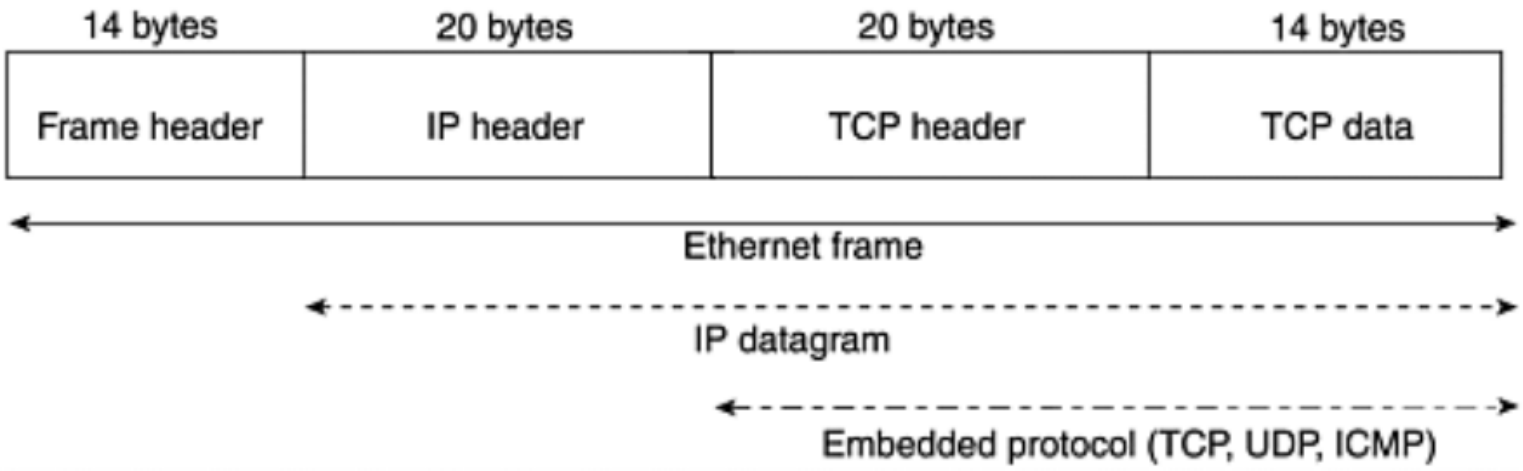


Headers are prepended as the packet descends the stack

Stephen Northcutt, Network intrusion detection book, figure 1.4, page 12

Sample packet

Figure 2.1. Sample packet.



Exercise

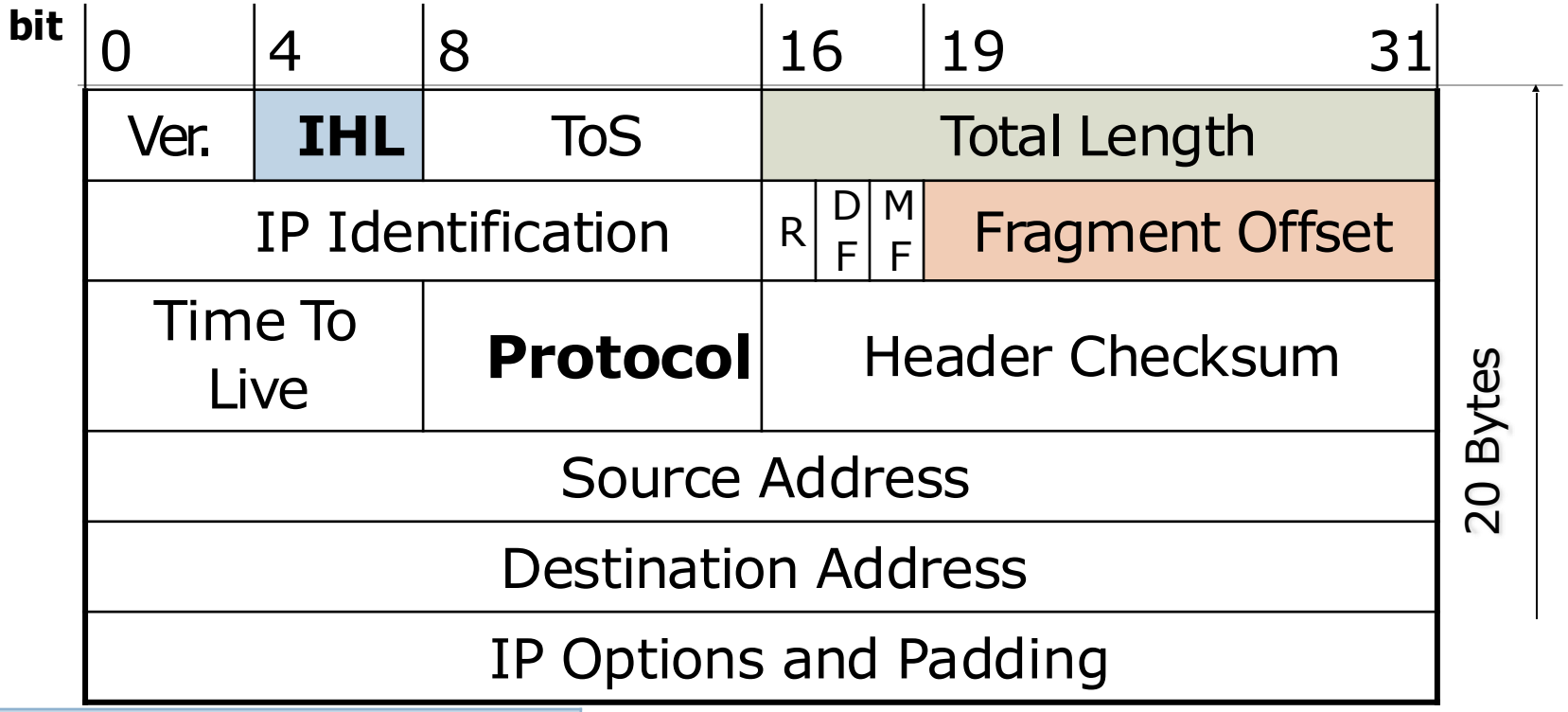
Open Help Sheet on DLE

menti

Notations

- 1 word is equal to 4 bytes
- Byte offset (or offset) is used to locate a field in the IP header, TCP/UDP/ICMP header
- Byte offfset is counted from 0: offset 0, offset 1, …
- Size of 1 byte offset is 1 byte
- In the following examples, all values are presented in hexadecimal. Example: 4e is a hexadecimal value. If this is converted into binary, it is 01001110 (1 byte)

IPv4 Header fields



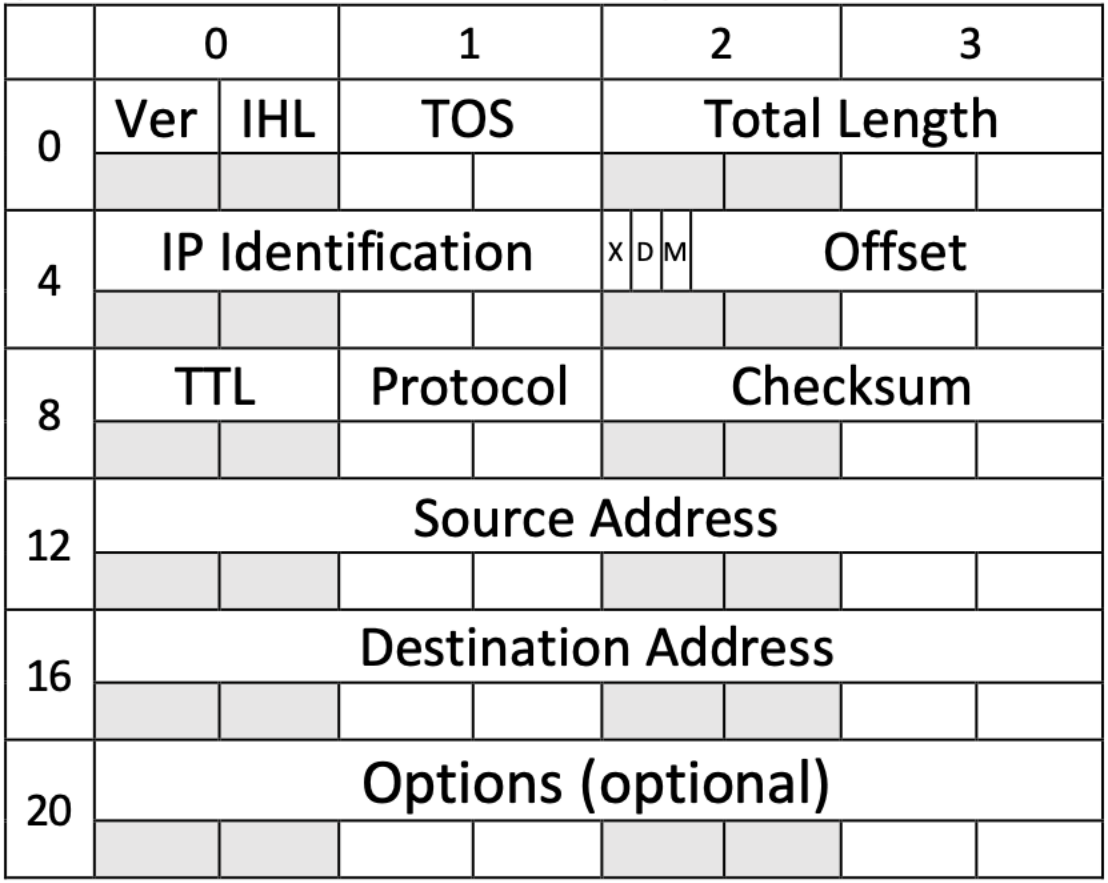
IHL (IP Header Length in words. Each word is 4 bytes)
Total Length
Fragment Offset

Multiply by 4 to convert to Bytes

No need to convert, already in Bytes

Multiply by 8 to convert to Bytes

Byte offset


0
1
2
3
0
Ver
IHL
TOS
Total Length
4
IP Identification
X
D
M
Offset
8
TTL
Protocol
Checksum
12
Source Address
16
Destination Address
20
Options (optional)

Filter IP header fields

1. Version: ip[0]&0xf0
2. IP Header Length: ip[0]&0x0f
3. ToS/Differentiated Services Byte: ip[1]
4. Flags: ip[6]

8	4	2	1	8	4	2	1
X	D	M	O	O	O	O	O

X: Reserved, D: Do Not Frag. M: More Fragments

5. Fragment Offset: position of this ip datagram's payload in original packet (multiply by 8)
6. Protocol ip[9]

1	ICMP	17	UDP	50	ESP
2	IGMP	41	IPv6	51	AH
6	TCP	47	GRE	115	L2TP

Example

```
20:55:48:520619 192.168.143.5 >
192.168.143.101: icmp: echo request

4e00 004c 05d9 0000 4001 d970 c0a8 8f05
c0a8 8f65 4424 0d03 c0a8 8f05 0033 1828
c0a8 8f65 0000 0000 c0a8 8f65 0000 0000
c0a8 8f05 0000 0000 0800 21a2 bd02 0100
0033 1828 0000 0000 0000 0000
```

IP Header Length (IHL) is at offset 0: 14 32-bit words = 14 *4 = 56 Bytes

IP header value is the underlined in the above example

Protocol (at offset 9): 1 means ICMP

IP options is from offset 20 to 55

IP flags

DF = Don’t Fragment. This flag is to indicate to fragment the diagram (if necessary) or not

MF = More Fragment. This flag is to indicate if more fragments follows the current one.

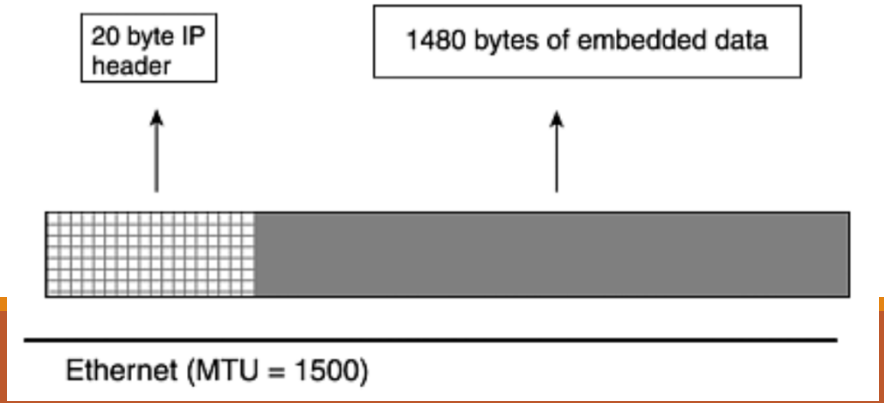
Fragmentation Offset

- Fragmentation offset (13-bit field) includes a part of offset 6 and offset 7
- It is used to indicate the starting position of the data in the fragment in relation to the start of the data in the original packet. This information is used to reassemble the data from all the fragments.
- To convert to bytes, multiply the value of “fragmentation offset” by 8

Fragmentation theory

- The maximum transmission unit (MTU) is the largest size frame/packet (in bytes) that can be transmitted across a data link.
- The standard size MTU for Ethernet is 1500 bytes excluding the Ethernet header.
- When a datagram is larger than 1500 bytes: if the flag DF=0, it is broken into fragments of 1500 bytes or less. If DF=1, the datagram is not broken and the router will discard the diagram.
- The ”original” IP header will be cloned for all the fragments. It contains the identical fragment identification number

Figure 3.1. Ethernet datagram packaging.



Example

A datagram has 4028 bytes (> MTU (1500 bytes)) is broken into 3 fragments (1st and 2nd fragment has 1500 bytes each, the final one has 1068 bytes). All fragment has a similar IP header

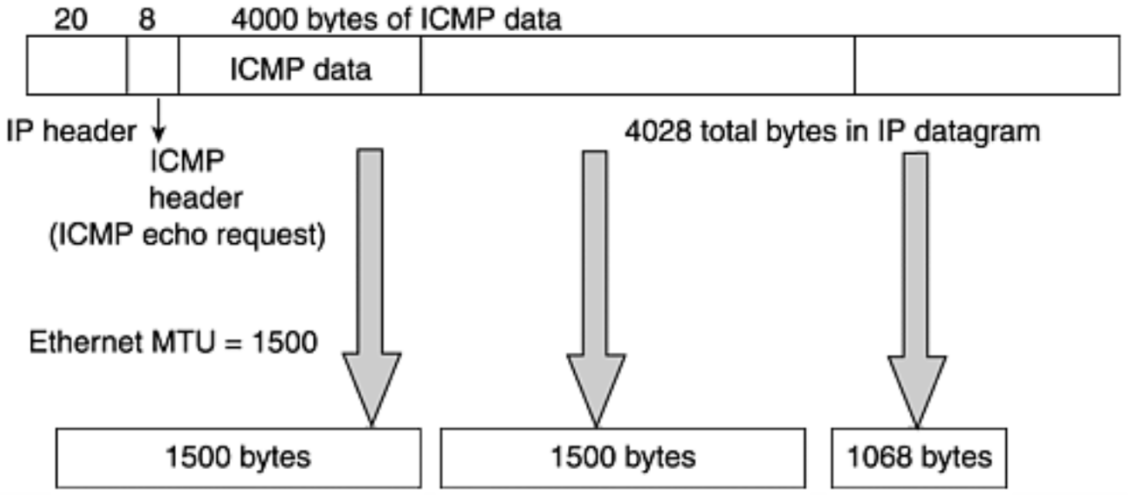
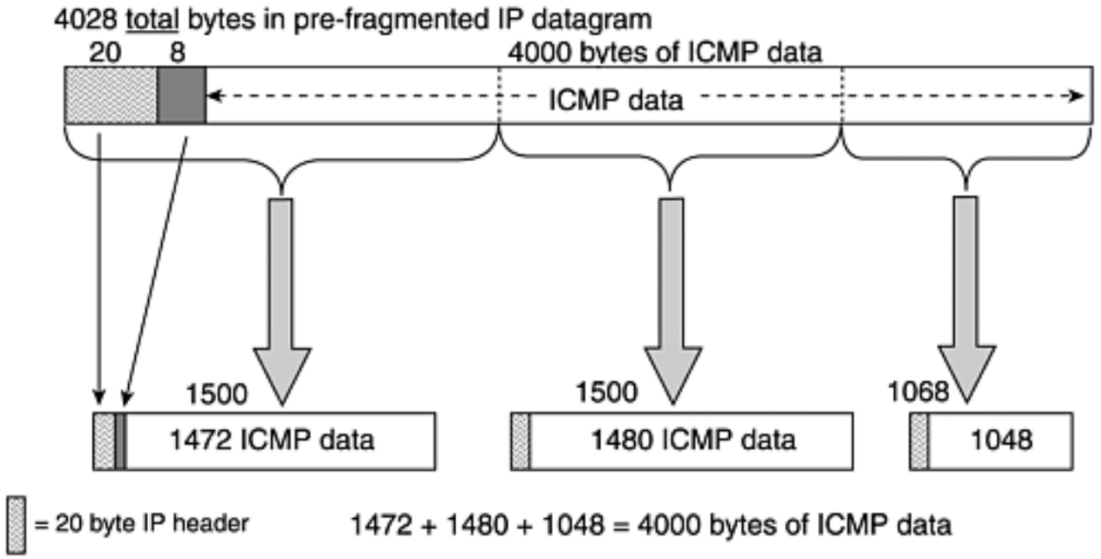
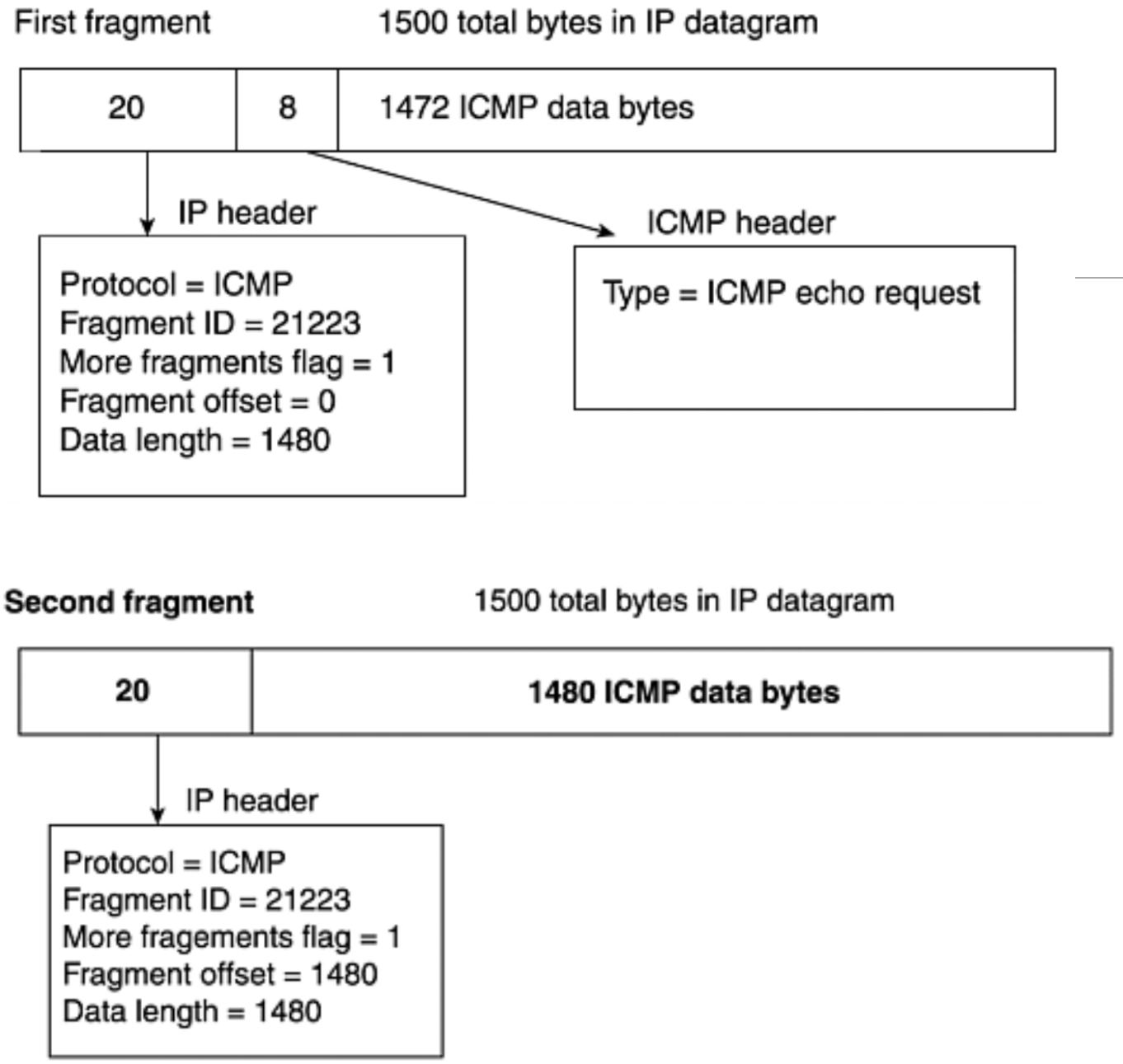
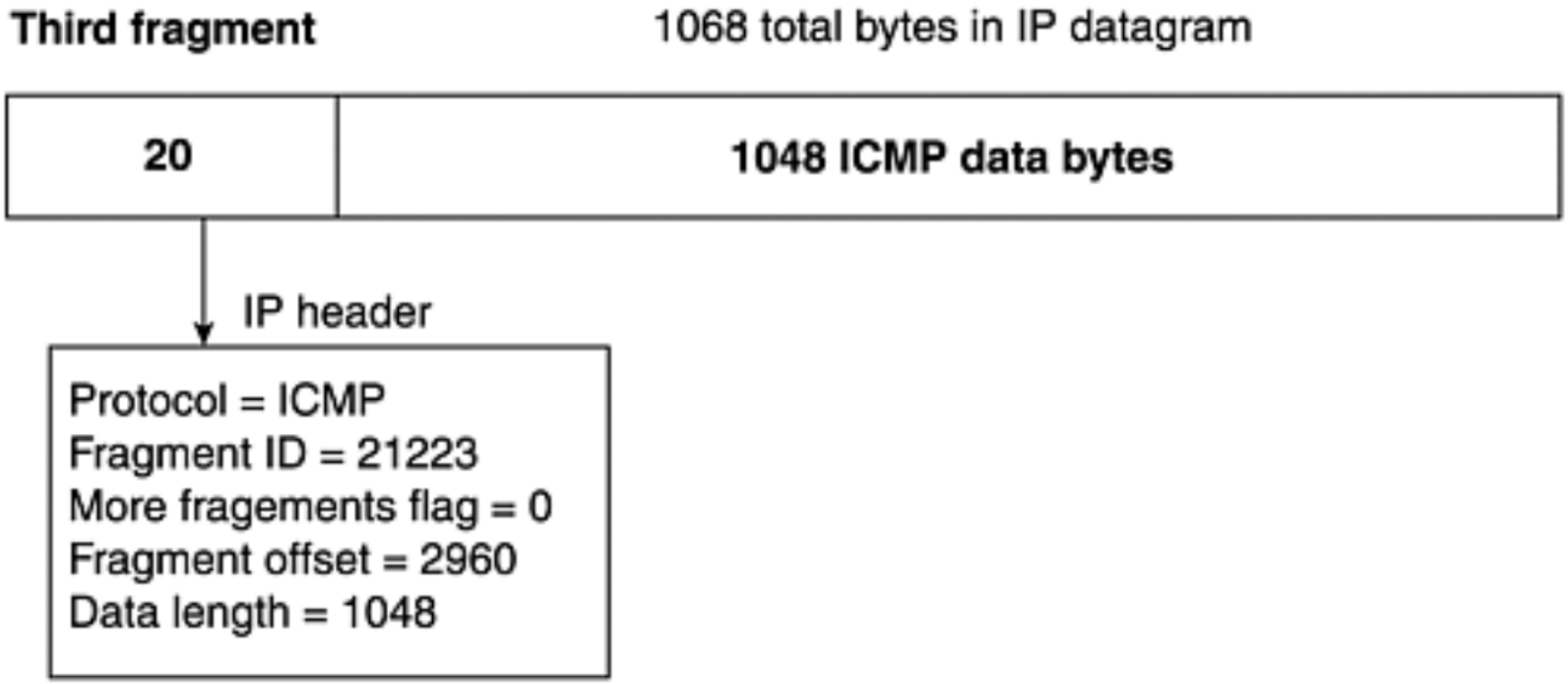


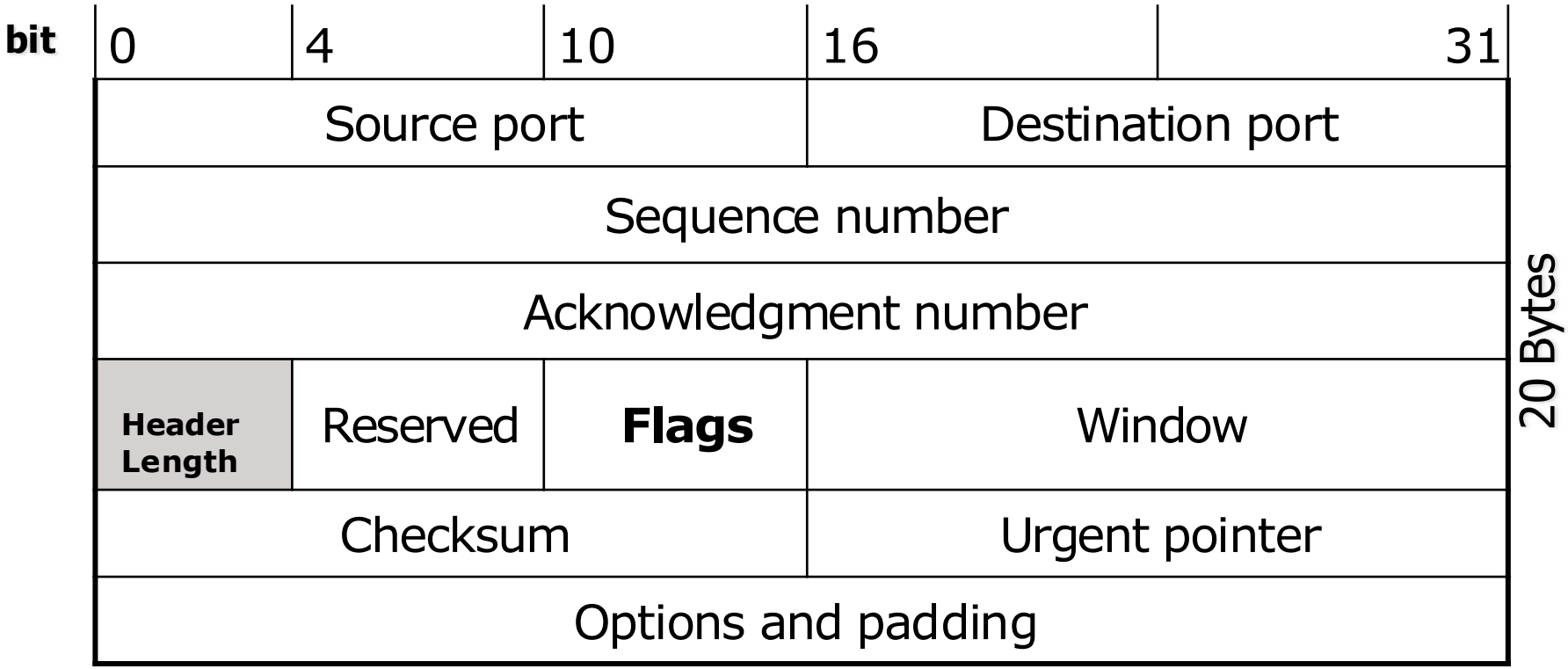
Figure 3.3. Byte allocations per fragment.




First fragment
1500 total bytes in IP datagram
20
8
1472 ICMP data bytes
IP header
ICMP header
Protocol = ICMP
Fragment ID = 21223
More fragments flag = 1
Fragment offset = 0
Data length = 1480
Type = ICMP echo request
Second fragment
1500 total bytes in IP datagram
20
1480 ICMP data bytes
IP header
Protocol = ICMP
Fragment ID = 21223
More fragements flag = 1
Fragment offset = 1480
Data length = 1480

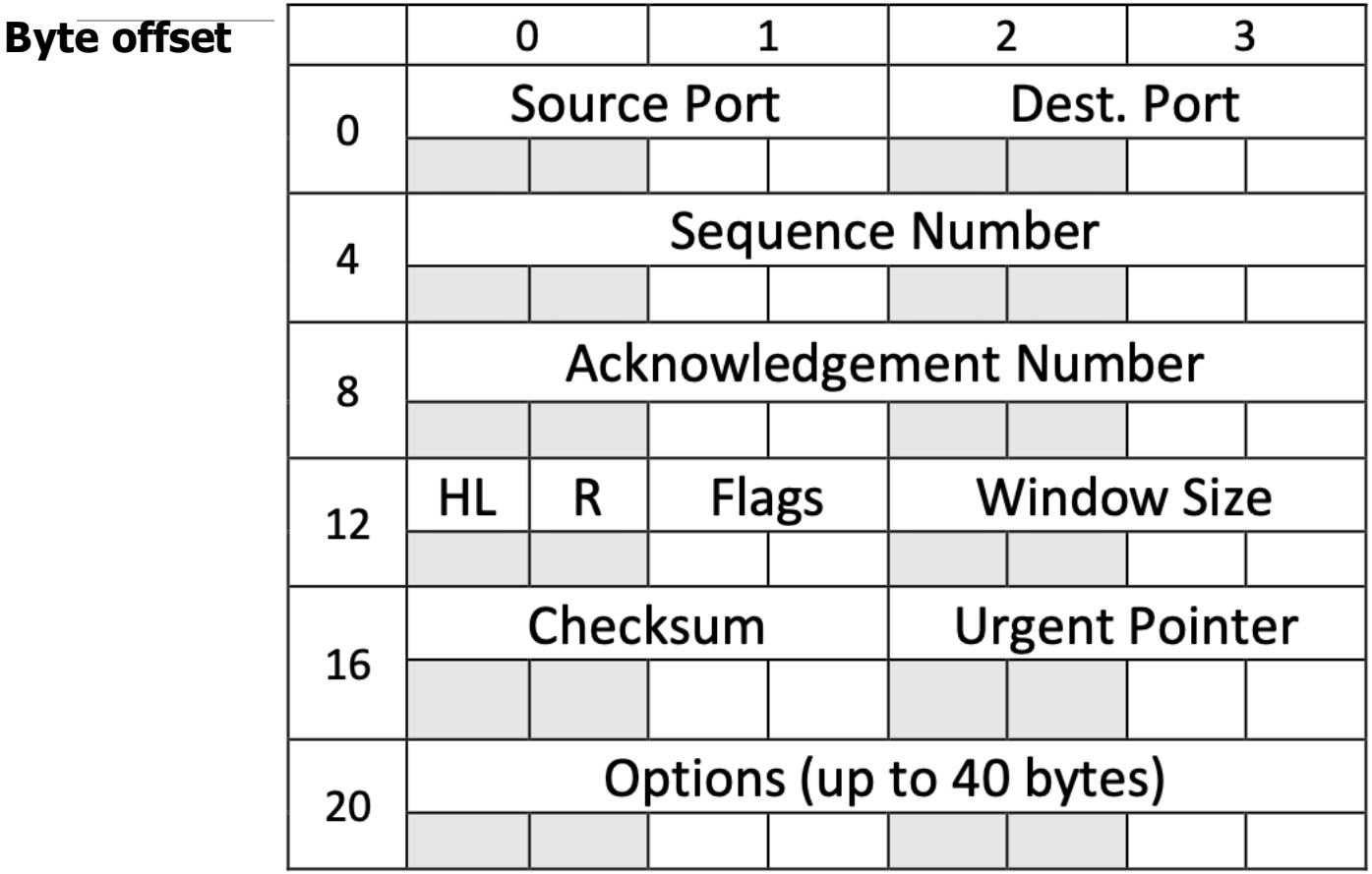

Third fragment
1068 total bytes in IP datagram
20
1048 ICMP data bytes
IP header
Protocol = ICMP
Fragment ID = 21223
More fragements flag = 0
Fragment offset = 2960
Data length = 1048

TCP Header fields



Header Length Multiply by 4 to convert to Bytes

TCP


Byte offset
0
1
2
3
0
Source Port
Dest. Port
4
Sequence Number
8
Acknowledgement Number
12
HL
R
Flags
Window Size
16
Checksum
Urgent Pointer
20
Options (up to 40 bytes)

Filter TCP header fields

1. Sequence Number tcp[4:4]:
2. Ack. Number tcp[8:4]
3. TCP Header Length tcp[12]
4. TCP flags tcp[13]

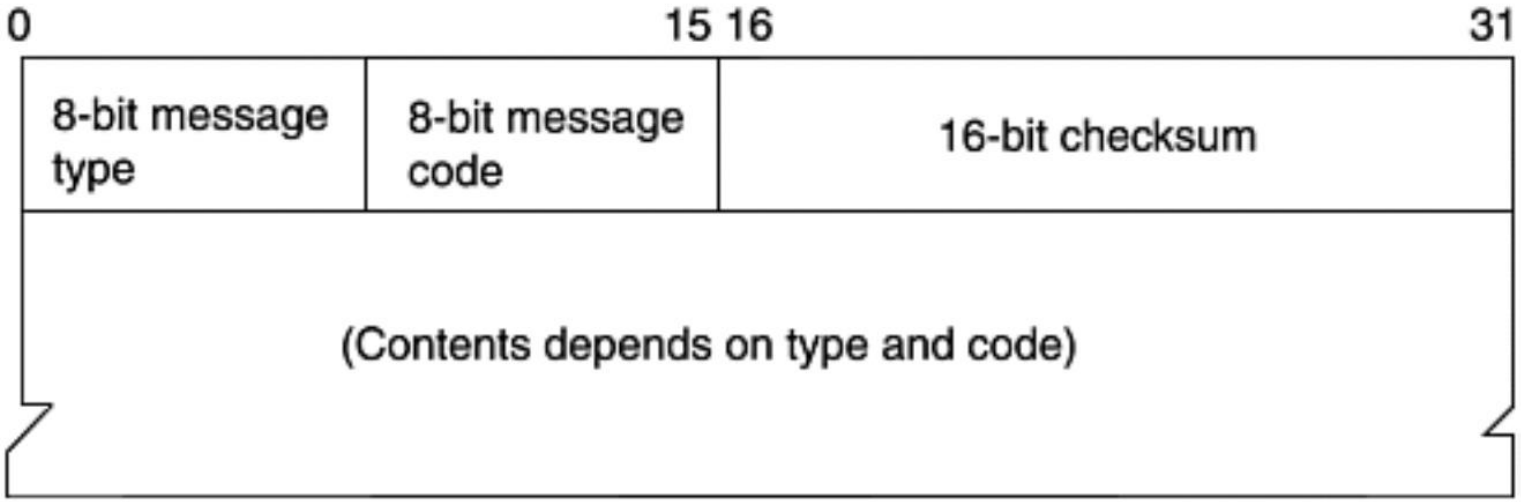
Common TCP Ports

20	ftp-data	80	http	443	https
21	ftp	88	kerberos	445	MS SMB
22	ssh	110	pop3	465	SMTPS
23	telnet	113	authd	1433	MS SQL
25	smtp	119	nntp	3128	Squid
43	whois	143	imap	3306	Mysql
53	dns	179	bgp	3389	MS Term.

- Flags tcp[13]

8	4	2	1	8	4	2	1
CWR	ECE	URG	ACK	PUSH	RES	SYN	FIN

ICMP header layout



Type	Code	Message
0	0	Echo reply
8	0	Echo request

Dissect a packet: Stephen Northcutt, Judy Novak. Network Intrusion Detection, Third Edition page 142, fig 7.2

Byte offset

ICMP Echo Request/Reply (Ping)				
	0	1	2	3
0	Type	Code	Checksum	
4	ICMP ID		ICMP Sequence	

Dissecting a datagram in hex

- Identify embedded protocol
  - Found in **9th Byte offset** of IP header
  - Determine what **type of header/data follows the IP header**
- Identify where header(s) stop(s)
  - Examine **IP header length**
    - Based on that you can tell where the embedded protocol begins
  - Examine **embedded protocol header length**
    - Based on that you can tell where the payload begins

Dissection Example

```
4500 0054 f23b 4000 ff01 d121 cccc cccc
cccc cccc 0000 9f00 d646 0000 b4cb 863a
56af 0e00 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
3435 3637 0000 4e00
```

- Embedded protocol, 9th Byte offset: 1 (ICMP)
- IP header length, 0 byte offset: 5 32-bit words =5*4 =20 Bytes
- ICMP header starts at 20th Byte offset: 0000
- In ICMP header: ICMP Type 0 Byte offset, ICMP code 1st Byte offset:

ICMP Type: 0 (ICMP echo reply), Code: 0

Your Turn – Exercise 1

```
4500 0054 b02f 0000 4001 122e 01f0 8f05
01f0 8f65 0800 836a cf1b 0000 ecec c339
0150 0900 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
```

Dissect the diagram to identify the following information: (menti)

- Embedded protocol
- IP header length
- Where does the embedded protocol header start?
- Any other information about the embedded protocol?

Your Turn

```
4500 0054 b02f 0000 4001 122e 01f0 8f05
01f0 8f65 0800 836a cf1b 0000 ecec c339
0150 0900 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
```

- Embedded protocol, 9th Byte offset: 1 (ICMP)
- IP header length, 0 byte offset: 5 32-bit words =5*4 =20 Bytes
- ICMP header starts at 20th Byte offset: 0800
- ICMP Type 0 Byte offset, ICMP code 1st Byte offset:

ICMP Type: 8 (ICMP echo request), Code: 0

Exercise 2

4500 0030 df3c 4000 8006 633f 0102 0304

0403 0201 0b64 0015 48f3 05b1 0000 0000

7002 2000 50b6 0000 0204 05b4 0101 0402

Dissect the diagram to identify the following information: (menti)

- Embedded protocol
- IP header length
- Where does the embedded protocol header start?
- Any other information about the embedded protocol?

Exercise 2

4500 0030 df3c 4000 8006 633f 0102 0304

0403 0201 0b64 0015 48f3 05b1 0000 0000

7002 2000 50b6 0000 0204 05b4 0101 0402

- Embedded protocol at offset 9: 06 means TCP
- IP header length at offset 0, nibble 1: 5 means 5 words or 5*4=20 bytes
- Where does the embedded protocol header start? Offset 20
- Any other information about the embedded protocol?
  - Source port at offset 0 & 1 in TCP header: 0b64 in hexadecimal
  - Destination port at offset 2 & 3: 0015 in hexadecimal or $16^1 + 5*16^0$=21 in decimal => FTP service
  - TCP header length at high-order nibble of offset 12: 7 means 7 words or 7*4=28 bytes => there are TCP options

Filtering Network Traffic

TCPDUMP – MACROS – FIELDS – MASKS

Tcpdump Filters*

Tcpdump is a general-purpose packet sniffer and incident response tool

- Allow you to investigate records of interest
- Can filter any field (e.g. IP header length, TCP flags)
- Macros for most commonly used fields
  - E.g. "port" or "host"
- For less common fields, it is possible to identify the location of field in protocol header
  - E.g. ip[0], tcp[13]

*Tcpdump Filters: http://www.cs.ucr.edu/~marios/ethereal-tcpdump.pdf

Using macros

Syntax:
<macro> <value>
Ex: src port 25
(*src port* is macro, *25* is value)

Macro	Description
host	select the record if either the source or destination host matches IP address or hostname
net	select the record if either the source or destination subnet matches
port	select the record if either the source or destination port matches
src host	select the record if the source host matches IP address or hostname
dst host	select the record if the destination host matches
src net	select the record if the source subnet matches
dst net	select the record if the destination subnet matches
src port	select the record if the source port matches
dst port	select the record if the destination port matches
icmp	select the record if the protocol field in IP header* has a value of 1
tcp	Selects any TCP record (the protocol field in IP header* has a value of 6)
udp	Selects any UDP record (the protocol field in IP header* has a decimal value of 17)

Macro	Description
tcp[tcpflags]	TCP flags field. The following TCP flags are available to compare with: tcp-syn TCP SYN (Synchronize) tcp-ack TCP ACK (Acknowledge) tcp-fin TCP FIN (Finish) tcp-rst TCP RST (Reset) tcp-push TCP Push
net	select the record if either the source or destination subnet matches
port	select the record if either the source or destination port matches

- tcpdump -r FILE: Reads and displays packets from a saved file.
- tcpdump -c COUNT: display first COUNT packets
- tcpdump -q: Quick output with brief information.
- tcpdump -e: Displays MAC addresses.
- tcpdump -A: Shows packet contents (payload) in ASCII format.
- Tcpdump -x: displays in hex
- tcpdump -xx: Displays link layer in hexadecimal format.
- tcpdump -X: Shows both hexadecimal and ASCII formats.
- Tcpdump -v: verbose (multiple v: more verbose)
- Tcpdump –n: do not resolve IP addresses / ports

Example

Use DG1 data (downloaded from DLE/ data lab 3)

1. tcpdump -r DG1 icmp (DG1 can be downloaded from DLE/ data lab 3)
2. tcpdump tcp
3. tcpdump tcp dst port 81
4. tcpdump " tcp[tcpflags]&tcp-syn!=0"
5. tcpdump "tcp[tcpflags] & (tcp-syn|tcp-ack) != 0" to capture TCP packets with at least the SYN (Synchronize) or ACK (Acknowledge) flags set.

Header: fields and position

(filter based on value of a field in a header)

Syntax:

<protocol header>[offset:length] <relation> <value>

(to identify fields >= 1 byte (e.g. udp[0:2]))

ip[9] = 1	Selects records with IP protocol of 1 (ICMP records). (*ip* is protocol header, *9* is offset, = is relation, *1* is the value)
tcp[2:2] < 20	Selects records with TCP destination port less than 20 (*tcp* is protocol header, *2* is offset, *2* is the length, < is relation, *20* is the value)
udp[6:2] != 0	Selects any udp record with non-zero UDP checksum
icmp[0] = 8	Selects any record with ICMP Type of 8 (echo request)

Filtering based on bits

What if a field is less than 1 Byte?

- Bit masking
- "AND" unwanted bits with 0's

Example: to identify the value of the IP header length, we use mask 00001111 (0x0F)

	4-bit version				4-bit header length			
1st Byte IP Header	0	1	0	0	0	1	0	1
AND								
MASK	0	0	0	0	1	1	1	1
Result	0	0	0	0	0	1	0	1

Masking the TCP Flag Bits

- What would be the mask to single out the ACK bit?

	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
TCP[13]	0	0	0	1	0	0	1	0
AND								
MASK	0	0	0	1	0	0	0	0
Result	0	0	0	1	0	0	0	0

tcp[13] & 0x10 != 0

Alternatively: tcp[13] & 0x10 = 0x10

Example

Isolate packets with specific TCP flags set.

tcpdump '**tcp[tcpflags] == tcp-syn**'

tcpdump 'tcp[13] & 2 != 0'

Exercise

Write a filter to discover any traffic that attempts to include data on the initial SYN packet.

Hint: use Helpsheet or SANS TCP/IP cheatsheet (downloaded on DLE)

Length of payload = Total length – IP header length - TCP header length

Total length, IP header length can be identified in IP headers

TCP header length can be identified in TCP header

Exercise

Use DG1 data (downloaded from DLE/data lab 3)

1. *How many packets use the ICMP protocol? UDP?*
2. *How many packets use the Telnet protocol?*
3. *Show both hexadecimal and ASCII format of udp packet*
4. *What is the IP address and MAC address of the host that sent*

Hint:

1. *use the ”wc -l” (wc means word count, -l to indicate to count lines) utility of linux to count the number of lines in the output*
2. *What is the port of Telnet service?*

Examination of Datagram fields

PACKET CRAFTING

“Crafting” packets

- Why are packets “crafted”?
  - Fingerprint remote operating system
  - Bypass detection by IDS
  - Send covert messages/channels
  - Elicit a response to map live hosts
- Packet crafting tools
  - E.g. nmap, hping, scapy
  - https://www.concise-courses.com/hacking-tools/packet-crafting-tools/

hping3

- Network tool that can send custom ICMP/ UDP/TCP packets and display target replies
- Easy to use syntax, provides some basic functionality
  - Can be used to test firewall rules
  - Portscans
  - Test network performance

https://github.com/Samsar4/Ethical-Hacking-Labs/blob/master/2-Scanning-Networks/1-hping3.md

Sample packet crafting

- Craft and send 1 packet:
  - Source IP 10.1.2.3
  - Destination IP 10.4.11.45
  - Source port 1024
  - Destination port 80
  - TCP SYN flag set
  - TCP window size 8192
  - TCP sequence number 12345

```
Hping3 -a 10.1.2.3 -s 1024 -p 80 -w 8192 -S -M
123456 -c 1 10.4.11.45
```

nmap

- Network Exploration tool and security scanner
- Scans hosts/networks to determine:
  - Live hosts
  - Running services
  - Operating system running on hosts

scapy

- Really powerful packet crafting tool
- Can manipulate all layers of TCP/IP stack
- Can specify application data (e.g. DNS)
- Can read, manipulate, write pcap files
- Command line API available for simpler tasks
- Scapy packages in python for more complex tasks

https://wiki.sans.blue/Tools/pdfs/ScapyCheatSheet_v0.2.pdf

Thank You