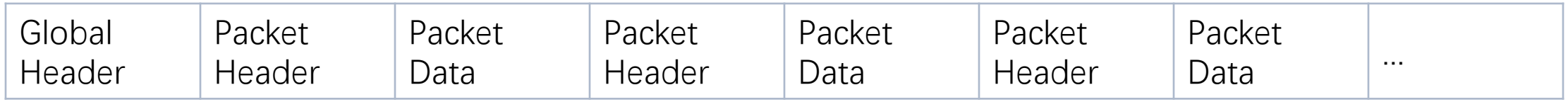
Tutorial 5: The Structure of PCAP Files

Objective: Know how to extract packets

http://www.tcpdump.org/linktypes.html

PCAP Format

- The file has a global header containing some global information followed by records for each captured packet, looking like this:



Global Header

```
typedef struct pcap_hdr_s {
        guint32 magic_number;   /* magic number  4 bytes*/
        guint16 version_major;  /* major version number 2 bytes */
        guint16 version_minor;  /* minor version number 2 bytes */
        gint32  thiszone;       /* GMT to local correction 4 bytes */
        guint32 sigfigs;        /* accuracy of timestamps 4 bytes */
        guint32 snaplen;        /* max length of captured packets, in octets 4 bytes */
        guint32 network;        /* data link type 4 bytes*/
} pcap_hdr_t;
```

- **magic_number**: used to detect the file format itself and the byte ordering. The writing application writes **0xa1b2c3d4** with it's native byte ordering format into this field. The reading application will read either **0xa1b2c3d4** (identical) or **0xd4c3b2a1** (swapped). If the reading application reads the swapped 0xd4c3b2a1 value, it knows that all the following fields will have to be swapped too.
- version_major, version_minor: the version number of this file format
- thiszone: the correction time in seconds between GMT (UTC) and the local timezone of the following packet header timestamps.
- sigfigs: in theory, the accuracy of time stamps in the capture; in practice, all tools set it to 0
- network: link-layer header type, specifying the type of headers at the beginning of the packet (e.g. 1 for Ethernet)

Packet Header

- typedef struct pcaprec_hdr_s {
- guint32 ts_sec; /* timestamp seconds */
- guint32 ts_usec; /* timestamp microseconds */
- guint32 incl_len; /* number of octets of packet saved in file */
- guint32 orig_len; /* actual length of packet */
- } pcaprec_hdr_t;

- ts_sec: the date and time when this packet was captured. This value is in seconds since January 1, 1970 00:00:00 GMT; If this timestamp isn't based on GMT (UTC), use *thiszone* from the global header for adjustments.
- ts_usec: in regular pcap files, the microseconds when this packet was captured, as an offset to *ts_sec*.
- incl_len: the number of bytes of packet data actually captured and saved in the file.
- orig_len: the length of the packet as it appeared on the network when it was captured. If *incl_len* and *orig_len* differ, the actually saved packet size was limited by *snaplen*

How to split packet

Global Header 24 bytes	Packet Header 16 bytes	Packet Data ? bytes	Packet Header	Packet Data	Packet Header	Packet Data	…

- Open the given pcap file in the binary mode, f = open(filename, “rb”)
- Read the first 24 bytes to get the global header, global = f.read(24)
- Check magic_number to determine the byte ordering (endianness),

Check thiszone to determine the time zone, etc

- Read the next 16 bytes to get the first packet header ph1 = f.read(16)
- Check incl_len for the length of packet, and check ts_sec for the time
- Read the next incl_len bytes for the first packet data, pd1 = f.read(incl_len)
- Continue the above steps to split every packet

Packet Data

Ethernet Header	IPv4 Header	TCP/UDP Header	Payload

Ethernet Header

bytes	Name
6	Destination MAC address
6	Source MAC address
2	Ethernet Type

IPV4 Header

4 bits	IP Version Number (4)
4 bits	IHL (IP HEADER LENGTH)
8 bits	Type of Service
16 bits	Total Length
16 bits	Identification
4 bits	Flags
12 bits	Fragment Offset
8 bits	Time to Live
8 bits	Protocol
16 bits	Header Checksum
32 bits	Source Address
32 bits	Destination Address

TCP header

TCP Header

Offsets Octet		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size													
16	128	Checksum																	Urgent pointer (if URG set)													
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

How to decapsulation for a packet data?

Ethernet Header 14 bytes	IPv4 Header Usually 20 bytes	TCP/UDP Header 20 – 60 bytes	Payload

- Assume we have obtained a binary string for a packet data by pd1 = f.read(incl_len) as explained in slide 7
- The Ethernet header is of fixed size, so it is easy to extract
- For IPv4 header, check IHL for the header length
- For the TCP header, the first 20 bytes are fixed, which contain the information we needed. Use data offset field to determine where the TCP header ends