CS-349 Networks Lab
Assignment 2: Analysing network packets using Wireshark

NAME: Mayank kumar Agrawal (150101033)

APPLICATION: Vimeo

1) List of all protocols used by the application vimeo at different layers:
Secure Socket Layer: TLSV1.2
Application Layer: HTTP
Presentation Layer: SSL
Transport Layer: TCP
Network Layer: IPv4
Physical Layer: Ethernet

a) Secure Socket Layer(SSL): Transport Layer security is a cryptographic protocol. It contains the content type which identifies the record layer protocol type contained in the record. The version field gives the current version, (here it is 1.2). The length gives the length of application data excluding the protocol header but including the MAC and padding trailers. Then the application data is present in encrypted form.

b) Application Layer (HTTP): A HTTP packet has a start line, header fields and an empty line. A client sends requests and the server sends responses. Start line has the request for the resource sent by the client and the status sent by the server. A header field different types of headers. A general header contains headers that are applicable to both request and response message. A request header contains headers that are applicable for only request message. A response header contains headers that are applicable for only response message. An entity header contains meta information about the entity body and, if no body is present, about the resource identified by the request. An empty line is used to indicate the end of the header field. It may contain an optional message body which carries the actual HTTP request data and HTTP response data.

c) Transport Layer (TCP): Firstly, a 16-bit source port is present followed by another 16-bit destination port. Then a 32-bit Sequence number is present which plays a dual role. If the SYN flag is set to 1, then the number present is the initial sequence number i.e. sequence number of the first data byte and then the ACK value is sequence number plus 1. But, if the SYN flag is set to 0, then it is the accumulated sequence number of the first data byte of this segment for the current session. Then a 32-bit acknowledgement number is present, if the value is set then this is the next sequence number that the sender is expecting. Then a 4-bit data offset is present. It specifies the size of the tcp header in 32-bits. A 3-bit Reserved is present and set to zero and is for future use. A 9-bit flag is present to denote various flag variables. A 16-bit window size is present after it which denotes the size of the receive window, which specifies the number of window size units that the sender of this segment is willing to receive. A 16-bit checksum is present for error-checking header and data. A 16-bit urgent pointer is present which if set, is an offset from the sequence number indicating the last urgent data type. A variable-sized options field is present whose length is determined by the data offset field. A padding is present to ensure that the TCP header ends and data begins on a 32-bit boundary. It is composed of zeroes.

d) Network Layer (IPv4): It has the version number (value is 4 in this case). Then it has the header length which denotes length of the entire header (in 4- bit which gives the number of 32-bit words). Then, Differentiated Services Code Point (DSCP) and Explicit Congestion Notification (ECN) is present. DSCP takes 6-bits whereas ECN takes 2-bits. Then a 16-bit field denotes the total length of packet including header and data. Then a 16-bit identification id present which helps in identifying different fragments of a single datagram. Then a 3-bit flag is present. Flags are used to determine whether IP packet is too large and if it can be fragmented or not. Then 13-bit fragment offset is present which tells the exact position of the fragment in the original IP packet. A 8-bit Time to live is present which helps in avoiding a loop in the network. The set

value is decremented by one when it crosses a hop. If it reaches zero then the packet is discarded. A 8-bit protocol number is present to determine at the network layer which protocol the packet belongs to. A 16-bit header checksum is used to keep the checksum value of the entire header which is used to check if the packet received is error free or not. A 32-bit address to give source address is present and then another 32-bit address is present to give destination address. And then there is another 32-bit options field which is optional and is used when header length is greater than 5. These may contain values for options such as Security, Record Route, Time stamp, etc.

e) Data Link Layer (Ethernet 2): A packet contains two mac addresses of 6 bytes each. There is a destination mac address and followed by the source mac address. Then there is 2 byte to denote type. It gives the type of protocol in the next layer. In a ethernet there is also a preamble (present in the beginning of the packet) and FCS(present in the end of the packet) present but since ethernet hardware filters them out they are not available to wireshark.

2. The values for various filed of the protocols:

a) Data Link Layer (Ethernet):

Since data link layer has access to hardware address. Therefore, we can see mac addresses of destination and source. Each is a 6-byte long number. The type denotes the protocol type in the next layer.

```
Ethernet II, Src: OneplusT_f3:2f:42 (c0:ee:fb:f3:2f:42), Dst: HonHaiPr_bf:d1:71 (70:77:81:bf:d1:71)
  Destination: HonHaiPr_bf:d1:71 (70:77:81:bf:d1:71)
      Address: HonHaiPr_bf:d1:71 (70:77:81:bf:d1:71)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: OneplusT_f3:2f:42 (c0:ee:fb:f3:2f:42)
      Address: OneplusT_f3:2f:42 (c0:ee:fb:f3:2f:42)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

b) Network Layer (IPv4):

IPv4 shows a lot of information about the IP packet in the format given above. We see the source IP as CSE Lab and destination is the proxy server of 2104 batch. We have visible fields like version (4) and total length of packet (60). The time to live is 60. The tcp protocol with protocol number 6 is present. The checksum is present. There are flag bits associated with the packet such as reserved bit and fragmentation.

```
Internet Protocol Version 4, Src: 157.240.185.17, Dst: 192.168.43.219
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 86
    Identification: 0x4fd3 (20435)
  Flags: 0x02 (Don't Fragment)
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 88
    Protocol: TCP (6)
    Header checksum: 0x8f49 [validation disabled]
    [Header checksum status: Unverified]
    Source: 157.240.185.17
    Destination: 192.168.43.219
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

c) Transport Layer (TCP):

TCP fields gives source and destination ports. Here, 44351 is the port for vimeo's website and 3128 gives the port of our proxy server. Other details include windows size, segment length, checksum and sequence number. A large number of flags is present in the TCP header. Here, SYN flag is set.

```
Transmission Control Protocol, Src Port: 443, Dst Port: 56487, Seq: 1, Ack: 1,
   Source Port: 443
   Destination Port: 56487
   [Stream index: 0]
   [TCP Segment Len: 0]
   Sequence number: 1    (relative sequence number)
   Acknowledgment number: 1    (relative ack number)
   1000 .... = Header Length: 32 bytes (8)
 > Flags: 0x010 (ACK)
   Window size value: 63
   [Calculated window size: 63]
   [Window size scaling factor: -1 (unknown)]
   Checksum: 0xb426 [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
 > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), SACK
```

d) Application Layer (HTTP):

HTTP gives the host name i.e. 'vimeo.com'. It also gives information about proxy connection and authorization. Other details are about user-agent (application process responsible for the packet), etc.

```
▼Transmission Control Protocol, Src Port: 44351 (44351), Dst Port: ndl-aas (3128), Seq: 0, Len: 0
  Source port: 44351 (44351)
  Destination port: ndl-aas (3128)
  [Stream index: 20]
  Sequence number: 0    (relative sequence number)
  Header length: 40 bytes
 ▶Flags: 0x002 (SYN)
  Window size value: 29200
  [Calculated window size: 29200]
 ▶Checksum: 0x42ff [validation disabled]
 ▼Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▶Maximum segment size: 1460 bytes
  ▶TCP SACK Permitted Option: True
  ▶Timestamps: TSval 38185907, TSecr 0
  ▶No-Operation (NOP)
  ▶Window scale: 7 (multiply by 128)
```

e) Secure Socket Layer:

Here, Content type is application data with TLS v1.2 and the length of application data is 2783 and then application data is present in encrypted from.

```
Secure Sockets Layer
 ⌄ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 41
      Encrypted Application Data: 3a6524cfe708004b01ce73f89baacdf7c4f6324464050b38...
```

3). There is Handshaking sequence in the connection.

Establishment of a connection:

```
5 0.456351    192.168.43.219    151.101.192.217    TCP    66 57082 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460…
6 0.477777    151.101.192.217   192.168.43.219     TCP    66 443 → 57081 [SYN, ACK] Seq=0 Ack=1 Win=28800 Le…
7 0.477969    192.168.43.219    151.101.192.217    TCP    54 57081 → 443 [ACK] Seq=1 Ack=1 Win=16384 Len=0
```

a) First, the client "192.168.43.219" sends a SYN segement. It is a request from the client for the server to synchronize.

b) Then, the server "151.101.192.217" sends an ACK and SYN segment. This is basically an acknowledgement from the server for the client's request. And a request from the server to the client to synchronize.

c) Then, client sends an ACK segment to acknowledge the server's request for synchronization. Then, a synchronized connection is established between the client and the server.

Termination of a connection:

192.168.43.219	52.222.139.189	TCP	54 57096 → 443 [FIN, ACK] Seq=1111 Ac…
52.222.139.189	192.168.43.219	TCP	54 443 → 57096 [FIN, ACK] Seq=5305 Ac…

a) The client first sends an ACK and FIN segment. The FIN segment denotes that no more data is to be sent and ACK is just the acknowledgement of the connection they have.
b) Then the server too sends an ACK and a FIN. The ACK is the acknowledgement of the FIN sent by the client and the FIN is denote that no more data is to be sent. This happens because they are independent connections.
c) Finally, the client sends an ACK to acknowledge the FIN sent by the client.

Upload:

192.168.43.219	151.101.192.217	TCP	55 [TCP segment of a reasse…
146.148.35.241	192.168.43.219	TLSv1.2	270 Application Data
151.101.192.217	192.168.43.219	TCP	66 443 → 57151 [ACK] Seq=1 …
192.168.43.219	151.101.192.217	TCP	1454 57151 → 443 [ACK] Seq=2 …
192.168.43.219	151.101.192.217	TLSv1.2	104 Application Data
192.168.43.219	151.101.38.109	TLSv1.2	134 Application Data
192.168.43.219	151.101.38.109	TLSv1.2	452 Application Data
192.168.43.219	151.101.38.109	TLSv1.2	134 Application Data
192.168.43.219	151.101.38.109	TLSv1.2	454 Application Data
192.168.43.219	146.148.35.241	TCP	66 57275 → 443 [SYN] Seq=0 …
192.168.43.219	146.148.35.241	TCP	66 57276 → 443 [SYN] Seq=0 …
151.101.192.217	192.168.43.219	TCP	54 443 → 57151 [ACK] Seq=1 …
151.101.38.109	192.168.43.219	TCP	54 443 → 57173 [ACK] Seq=1 …
151.101.38.109	192.168.43.219	TCP	54 443 → 57173 [ACK] Seq=1 …
151.101.38.109	192.168.43.219	TLSv1.2	166 Application Data
151.101.38.109	192.168.43.219	TLSv1.2	143 Application Data
192.168.43.219	151.101.38.109	TCP	54 57173 → 443 [ACK] Seq=95…
151.101.192.217	192.168.43.219	TLSv1.2	1064 Application Data

Download:

When a video has to be downloaded, then first a tcp handshake is established. Once, a tcp handshake is established, a tls handshake is initiated to send data securely. Once tls handshake is complete, then data is securely sent through the tls connection. Once the entire file is downloaded, then a call to terminate the connection is called. Since, no more data is to be sent both independently call to terminate the connection. After a terminate handshake is complete the process is over.

TLS Handshake protocol:

386	109.081536	192.168.43.219	172.217.163.195	TLSv1.2	252 Client Hello
387	109.183839	172.217.163.195	192.168.43.219	TCP	54 443 → 57044 [ACK] Seq=1 Ack=199 Win=44032 Len=0
388	109.228617	172.217.163.195	192.168.43.219	TLSv1.2	1454 Server Hello
389	109.230422	172.217.163.195	192.168.43.219	TCP	1454 443 → 57044 [ACK] Seq=1401 Ack=199 Win=44032 Len=1400 [TCP segment of a reassembled PDU]
390	109.230424	172.217.163.195	192.168.43.219	TLSv1.2	523 Certificate, Server Key Exchange, Server Hello Done
391	109.230662	192.168.43.219	172.217.163.195	TCP	54 57044 → 443 [ACK] Seq=199 Ack=3270 Win=16384 Len=0
392	109.238370	192.168.43.219	172.217.163.195	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
393	109.346217	172.217.163.195	192.168.43.219	TLSv1.2	338 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
394	109.377584	192.168.43.219	172.217.163.195	TLSv1.2	687 Application Data
395	109.377980	192.168.43.219	172.217.163.195	TCP	1434 57044 → 443 [ACK] Seq=958 Ack=3554 Win=16128 Len=1380 [TCP segment of a reassembled PDU]
396	109.378027	192.168.43.219	172.217.163.195	TLSv1.2	95 Application Data
397	109.490353	172.217.163.195	192.168.43.219	TCP	54 443 → 57044 [ACK] Seq=3554 Ack=2379 Win=48128 Len=0
398	109.689665	172.217.163.195	192.168.43.219	TLSv1.2	1061 Application Data
399	109.689667	172.217.163.195	192.168.43.219	TLSv1.2	698 Application Data
400	109.689855	192.168.43.219	172.217.163.195	TCP	54 57044 → 443 [ACK] Seq=2379 Ack=5205 Win=16384 Len=0

- The client sends a "Client hello" message to the server, along with the client's random value and supported cipher suites.
- The server responds by sending a "Server hello" message to the client, along with the server's random value.

- The server sends its certificate to the client for authentication and may request a certificate from the client. The server sends the "Server hello done" message.
- If the server has requested a certificate from the client, the client sends it.
- The client creates a random Pre-Master Secret and encrypts it with the *public key* from the server's certificate, sending the encrypted Pre-Master Secret to the server.
- The server receives the Pre-Master Secret. The server and client each generate the Master Secret and *session keys* based on the Pre-Master Secret.
- The client sends "Change cipher spec" notification to server to indicate that the client will start using the new *session keys* for *hashing* and encrypting messages. Client also sends "Client finished" message.
- Server receives "Change cipher spec" and switches its record layer security state to *symmetric encryption* using the *session keys*. Server sends "Server finished" message to the client.
- Client and server can now exchange application data over the secured channel they have established. All messages sent from client to server and from server to client are encrypted using session key.

Play/Pause:

When a page is loaded, a tcp handshake is established. Here, HTTP connect happens as there is a proxy server. In this case, first client asks the proxy server to forward the TCP connecting to the desired connection. The proxy server then proceeds to make the connection on behalf of the client. Once proxy server establishes the connection, it continues to proxy the tcp stream to and from the client. After the initial connection which is HTTP, the proxy server simply proxies the established connection. After this HTTP request is sent to get the data of the page. Once the data of the page is received, a tls handshake protocol is initiated. After a tls handshake protocol is established, the data can be securely sent. The Application data is sent securely through the tls connection. The tcp uses the acknowledgement technique. The client sends ACK to the server and the server sends PSH and ACK packets to the client. It is used to communicate regarding how much data has been sent and what is to be sent. And additionally, used to communicate what data is incoming.

4). Different Protocols are used at different layers. They each perform a specific task which is essential for performing the functions:

a) HTTP: It is an application layer protocol which uses client request and server response protocol in the client-server model. It is a sequence of request-response transactions. It is used to get various resources from the server. We can see from the traces that we use HTTP/1.1 which uses a connection multiple times to get the resources after the page has been delivered. Media independent: It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content.

b) TCP: It is a communication service between application program and internet protocol. We have seen from the traces that it handles the handshaking which is essential for beginning and termination of the connection. It provides host to host connectivity and control. It requests for transmission if and when a packet is lost. It uses acknowledgement technique to ensure reliability of packet transfers. Acts as a intermediate, Congestion Avoidance Features, Providing Flow Control, Providing Reliability and Transmission Quality Services.

c) IPv4: It handles tasks in the network layer. The job description is to deliver the packets from source to destination solely on the basis of IP address in the packet headers. It defines the packet structure that encapsulates the data to be delivered. It also defines the addressing methods that are used to label the datagram with source and destination information. It helps in identifying hosts and providing a logical location service.

d) Ethernet: It is the most common local area network technology in today's era. It works at the data link layer. Data regarding the mac addresses of source and destination are found in ethernet header. Data is sent through Ethernet in frames and is visible to us in wireshark. The frames also contain error checking capabilities to find damaged frames and discard them.

e) SSL: It is used to secure the data transmitted over the internet between the client and the destination servers. Encryption used is AES with 128 bit key in GCM mode. Handshaking: When the connection starts, the record encapsulates a "control" protocol—the handshake messaging protocol (*content type* 22). This protocol is used to exchange all the information required by both sides for the exchange of the actual application data by TLS.

5). Traces drive link: https://drive.google.com/open?id=1Y42p8TRkd4uiMK2T6SIxdD92gNJBwQj8

(Using lab proxy)

Sample Number	Sample 1	Sample 2	Sample 3
Host A	172.16.114.32:47481	172.16.114.32:47608	172.16.114.32:47637
Host B	202.141.80.22:3128	202.141.80.22:3128	202.141.80.22:3128
Throughput (A -> B)	3.028 KB/sec	2.553 KB/sec	1.962 KB/sec
Throughput (B -> A)	289.822 KB/sec	268.519 KB/sec	183.374 KB/sec
Avg packet size (A -> B)	77.895 bytes	81.548 bytes	85.386 bytes
Avg packet size (B -> A)	6159.197 bytes	7791.247 bytes	7142.370 bytes
No of packets lost	0	0	0
No of TCP packets (A -> B)	1342	1086	926
No of TCP packets (B -> A)	1645	1216	1047
No of responses/request	1.226	1.119	1.130
RTT	0.415 ms	0.433 ms	0.426 ms

6).

The webpage 'www.vimeo.com' has the IP address 151.101.128.217. However, for streaming video it is loaded from 'i.vimeocdn.com' having the IP address 151.101.85.133 . For uploads, the video is uploaded to 'fresnel.vimeocdn.com' having the IP address 151.101.38.109 . For downloads, the video is downloaded from '16-lvl3-pdl.vimeocdn.com' having the IP address 52.222.139.189. We have pinged the servers multiple times and at different times of the day.

Here, we see that upload and streaming are from the same hosted network but download is on a different network. The reason could be that any video on the website can be watched and all videos have been uploaded by the users. Whereas only a class of videos can be downloaded from the website. One server is present for viewing and uploading videos for everyone to view. Another server hosts the videos that can be downloaded. The reason for multiple sources can be to balance the page requests and serve the end user faster.