**A logo of a university of windsor

Description automatically generated with low confidence**

|  |
| --- |
| **Course Name** |
| Networking and Data Security (COMP-8677) |

|  |
| --- |
| **Document Type** |
| Lab 2 Work |

|  |
| --- |
| **Professor** |
| Dr. Shaoquan Jiang |

|  |
| --- |
| **Team - Members                               Student ID** |
| Harbhajan Singh                                 110100089 |

**Ques 1. Command dig is used to find the ip address of a hostname such as www.google.com. To do**

**this, you can simply run dig www.google.com. You might see the following result.**

Screenshot 0

We can see that www.google.com has an ip address - **172.217.13.164**

**(a). Try $ dig www.example.net to find out its ip address.**

**Ans 1(a)** IP address of [www.example.net](http://www.example.net) is 93.184.216.34

Screenshot 1

**Ques 1(b) run Wireshark on your VM, then $ dig www.example.net and stop wireshark. Look at the DNS request packet (using filter DNS to find it easily), confirm that the transport layer protocol is UDP. What are the values of this UDP header (you need to first check the header fields learned in class)?**

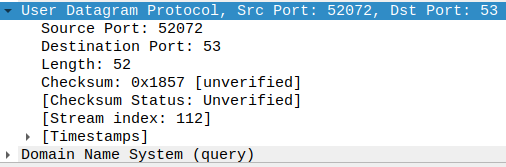
Ans 1(b) On looking at the DNS request packet(using filter DNS to find it easily), it is confirm that the transport layer protocol is UDP and The field values of UDP header are as follows:

Source Port: 52072

Destination Port: 53

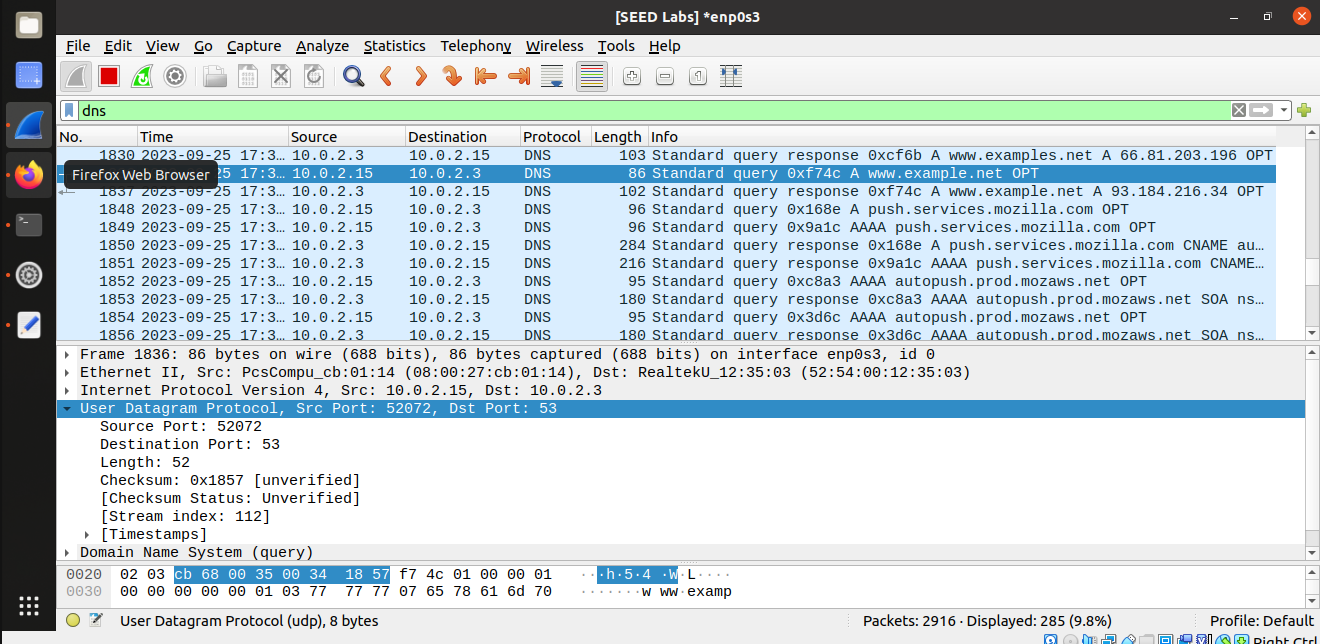
Length: 52

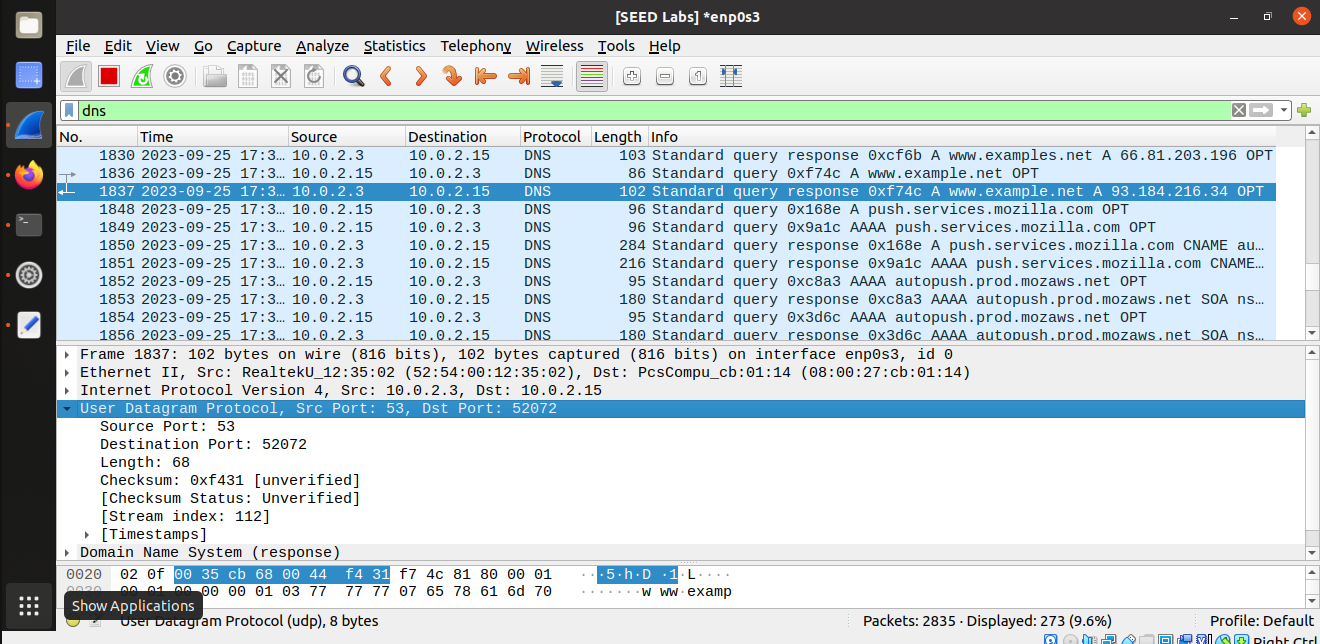
Checksum: 0x1857 [unverified]



This can be verified from the Screenshot 2, 3 and Screenshot 4.

Screenshot 2

Screenshot 3

Screenshot 4

**Ques 1(c) In the DNS request packet in step b, the destination IP is your local DNS server’s IP. What is this value? As said, DNS is serviced by UDP and has no connection setup before sending DNS request. You can confirm this by checking that there is no any packet in Wireshark exchanged between your VM and local DNS server, prior to the DNS request packet (show the screen shot of the window of Wireshark for the list of packets).**

**Ans 1(c)** The destination IP is your local DNS server’s IP and its value is 10.0.2.3

Also, we examined that there was/were no packet(s) captured in Wireshark for any prior packets exchanged between our VM and the local DNS server by referring to packets #1836 and #1837 So in case there are no preceding packets, that indicates that the DNS requests are sent independently without a prior connection. We can observe from the packet numbers (#1836 and #1837) as well from Screenshot 5 and 6.

Screenshot 5

Screenshot 6



**Ques. 2 Run Wireshark and then access www.example.net using Firefox (you might need to clear the browser history). Then stop the Wireshark. Check your list of packets in Wireshark window, filtered by the ip address of www.example.net. You can see that before the HTTP request to www.example.net, there is a connection stage with three packets: SYN packet, SYN-ACK packet and ACK packet. This is to provide the connection setup between your VM and www.example.net. Confirm this. Also, confirm that the transport layer protocol in these packets (check one of them is good enough) is TCP. When the message exchange starts, you can see ACK packet. This is to confirm the receipt of a packet. Find out such a packet. This is to find a packet with flags bit A=1. This provides an evidence that TCP is a reliable protocol. This is different from the UDP protocol. ACK packet might or might not contain the application data. Verify the ACK packet you consider (any of them is ok) to see if it contains application data.**

**Ans. 2**



When we accessed **www.example.net** using **Firefox** and examined the packet data in the Wireshark window, filtered by IP address of **93.184.216.34** which is of website (**www.example.net**). We noticed an initial connection setup stage before the HTTP request to the website. This setup involved three packets: a SYN packet, a SYN-ACK packet, and an ACK packet (as highlighted in **Screenshot 7**). These packets facilitated the establishment of a connection between our VM and www.example.net. All these packets used the TCP protocol for transport.

As the message exchange commenced, we also observed ACK packets confirming the receipt of data. These ACK (Packet with flag bit A=1) packets play a crucial role in indicating the reliability of the TCP protocol, as shown in **Screenshot 8**.

Starting from **packet Number 8** in **Screenshot 8**, we identified a specific ACK packet traveling from **source IP address 10.0.2.15** to **destination IP address 93.184.216.34**. This packet traveled from **source port 38222** to **destination port 80**, which is commonly associated with HTTP traffic. It's essential to note that ACK packets themselves do not carry application data. Instead, they serve to acknowledge the receipt of data from previous TCP segments. In our case, this **ACK packet (Ack= 309312002)** acknowledges the reception of data with a **sequence number of (Seq=3849216195)** and **header length 20 bytes**. While the ACK packet has a **length of 54 bytes**, it does not contain any application data.

In the TCP protocol, ACK packets are primarily used to confirm successful data reception and manage data flow between the sender and receiver. They do not typically include application data, which is typically transmitted in separate data packets.

Screenshot 7

Screenshot 8



**Ques. 3. Run Wireshark and access www.example.net and then close your webpage and stop your Wireshark. Answer the following questions.**

1. **Find out the first packet from your VM to www.example.net (you should know the ip address of www.examplenet now). This should be the SYN-packet (i.e., the first packet of the 3- way handshake protocol). What is source port # and destination port #? Confirm that they are in the TCP header in the Wireshark packet window. What is source IP and destination IP? Confirm that they are in the ip header in the Wireshark packet window.**

**Ans. 3 (a).**



The packet labeled as No. 1 in the screenshot represents the initial communication from a virtual machine (using **Firefox**) to the website **www.example.net**. The IP address associated with **www.example.net** is **93.184.216.34**.

In this communication:

* The source port number used is **37978**.
* The destination port number is **80**.

We can also verify these details by examining the TCP header information in the Wireshark packet window, as shown in **screenshot 9**.

Screenshot 9



Based on the information in **screenshot 10**, we can verify that **the source IP (10.0.2.15)** and **destination IP (93.184.216.34)** are indeed present within the IP header displayed in the Wireshark packet window.

Screenshot 10

1. **Look at SYN-packet. What is the sequence #? It is a random number. Confirm this.**

**Ans 3 (b).**



The **sequence number**, which can be confirmed by referring to **screenshot 11**, is **4123990051**.

Screenshot 11

1. **Find out in the TCP header the flag bits U|A|P|R|S|F in the SYN-ACK packet.**

**Ans 3 (c).**

In the TCP header of the SYN-ACK packet, the flag bits corresponding to **U | A | P | R | S | F** in SYN – ACK packet are **0 | 1 | 0 | 0 | 1 | 0** respectively which means 'U,' 'A,' 'P,' 'R,' 'S,' and 'F' are set as follows: 'U' is not set, 'A' is set to 1, 'P' is not set, 'R' is not set, 'S' is set to 1, and 'F' is not set (Refer to **Screenshot 12**).

To simplify, only the 'A' and 'S' flag bits are **activated** in this SYN-ACK packet.

Screenshot 12

1. **The receive window field is to tell its partner the current receive-buffer size it has. Find out the window size of SYN-ACK packet and that of http response packet. Are they equal?**

**And 3 (d).**

The **SYN-ACK packet** and the **HTTP** **response packet** share the **same window size**, both being **65535**. This can be confirmed by referring to **Screenshot 13** and **Screenshot 14.**

Screenshot 13

Screenshot 14

1. **Find out the sequence # of http request packet and its payload size (the segment len is the payload size). The next sequence # is the sum of these two numbers. Verify that this is indeed the sequence # of the next packet sent by your VM.**

**Ans 3 (e).**

**Refering Screenshot 15 –**

In the context of the HTTP Request Packet (**Frame 4**), we observe a sequence number of **4123990052** and a payload size (segment length) of **444 bytes**. To determine the next sequence number, we apply the formula:

**Next Sequence Number = Current Sequence Number + Payload Size**

Next Sequence Number = 4123990052 + 444 = 4123990496

This calculation allows us to find the next sequence number, which in this case is **4123990496**.

Screenshot 15

Let's now verify the sequence number of the next packet sent from our virtual machine. The packet capture data indicates the details for the upcoming packet (**Frame 6**), as shown in **Screenshot 16**.

Screenshot 16

Sequence number of Frame 6 -: **220928002 (Screenshot 16)**

The Acknowledgement Number of Frame 4 will become the Sequence Number of Frame 6. The sequence number for **Frame 6,** which is **220928002**, aligns perfectly with the previously calculated **next sequence number (4123990496)** derived from **Frame 4**. This confirmation establishes that the **sequence number in Frame 6** is in harmony with the calculated value based on the sequence number and payload size of the HTTP request packet in Frame 4.

1. **Find out the acknowledgement # in http response packet. Is this the same as the next sequence # you calculated above for the request packet? Explain why?**

**Ans 3 (f).**

**(Taking reference from the screenshots 15 and 16, and above calculation of answer 3(e)):**

In the analysis of the HTTP response packet (Frame 6) and by referring to the information in **screenshots 15 and 16**, we've noticed something interesting about the acknowledgment number. It appears to match the next sequence number that we previously calculated for the HTTP request packet (**Frame 4**), which is **4123990496**.

This alignment of acknowledgment and sequence numbers is a fundamental concept in TCP. In TCP, the acknowledgment number within a packet indicates the expected sequence number of the next byte. When the sender transmits data (like the HTTP request), it increments its sequence number for the subsequent packet. The receiver, in turn, confirms the receipt of data by specifying the next expected sequence number.

In our case, the fact that the acknowledgment number in the HTTP response packet (**Frame 6**) matches the next sequence number calculated for the HTTP request packet (**Frame 4**) signifies that the server has acknowledged the complete receipt of the HTTP request packet. Furthermore, it expects the next packet (if any) to carry a sequence number of **4123990496**.

This consistency in sequence and acknowledgment numbers lies at the core of TCP's reliability and flow control mechanisms. It ensures that data is transmitted accurately, received in the correct order, and contributes to the efficiency and dependability of data transfer over TCP connections.

1. **What is the flags bits U|A|P|R|S|F in the http response packet?**

**Ans. 3 (g).**

The HTTP response packet includes a set of flag bits of **U|A|P|R|S|F** in the HTTP response packet is **0 | 1 | 1 | 0 | 0 | 0**.

In this particular response, only the A and P flag bits are activated (**SET**), indicated by values 1 in their respective positions, signifying specific attributes of the response packet.

Screenshot 17



1. **Find out the packet your VM requests to terminate the TCP connection. This packet will be sent when you close the webpage. What is the flags bit U|A|P|R|S|F in this packet?**

**Ans. 3 (h).**

Packet #156, as shown in screenshot 18, serves as the termination request for the TCP connection, having corresponding flag bits of **U|A|P|R|S|F** in the http response packet is **0 | 0 | 0 | 1 | 0 | 0** which means only the 'R' flag bit **activated (SET)** among the U|A|P|R|S|F flag bits in the HTTP response packet.

Screenshot 18

**References**

1. **Week 2 – Class 2 Notes**
2. **Week 2 – Class 2 Instruction Document**