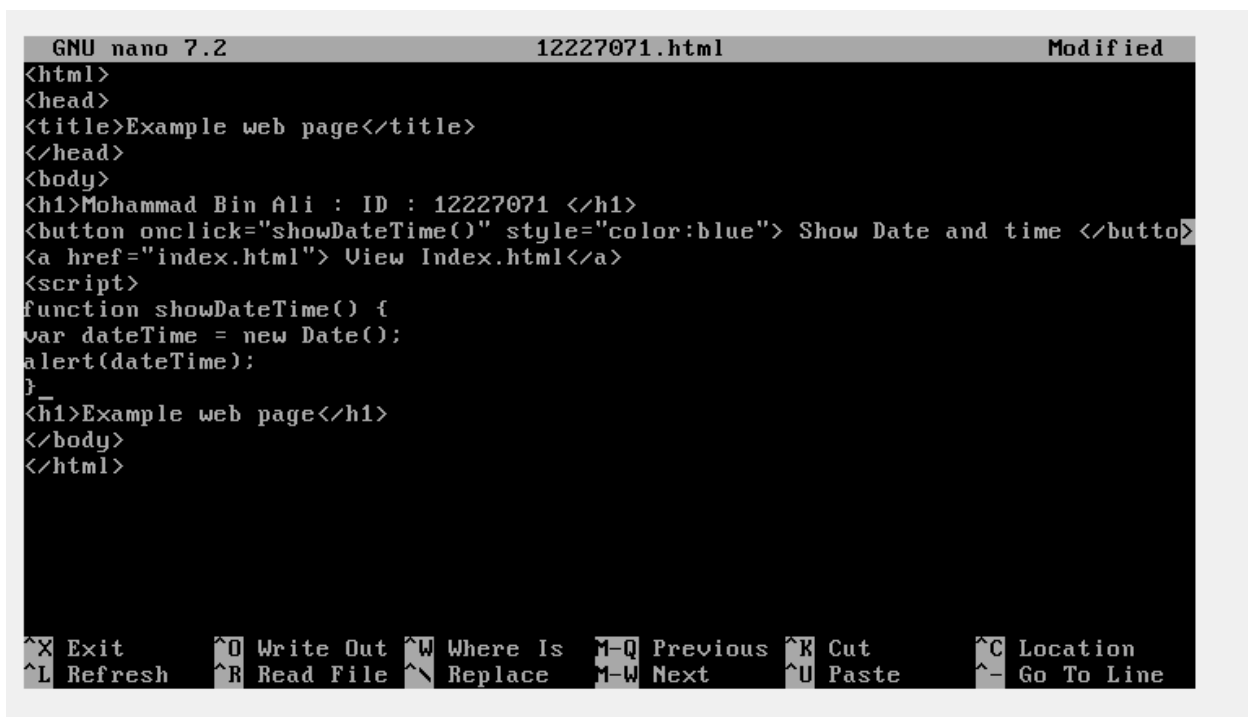


Week 5

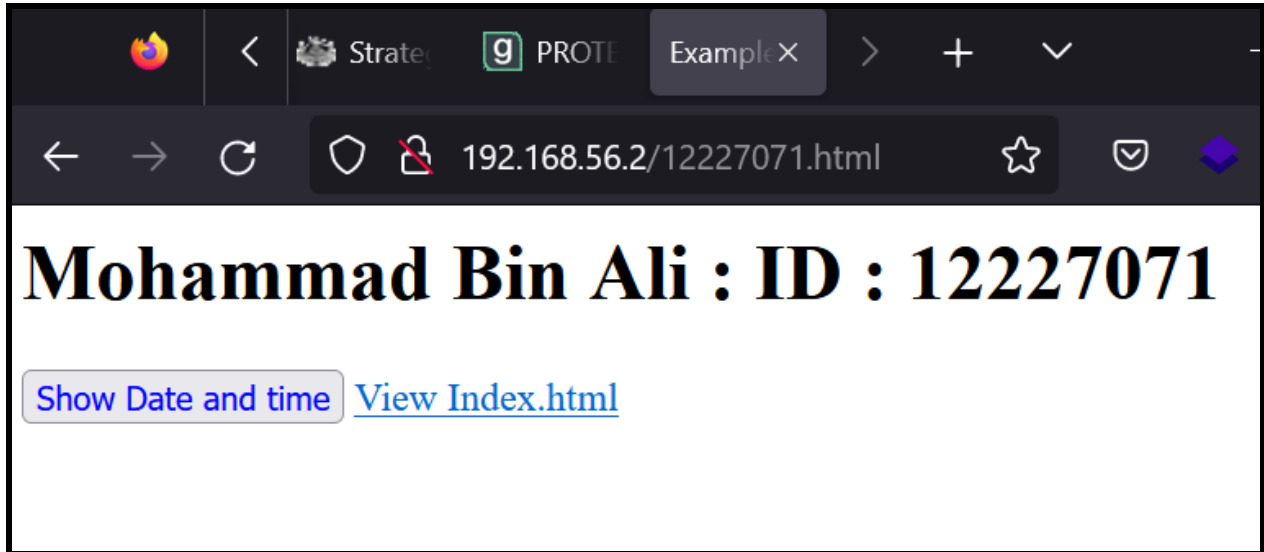
Task 1. Create Web Pages in OpenWRT

- Copy the index.html file to a new HTML file named by your student ID, e.g., 12345678.html.
- Add a link in index.html to the new HTML file.
- Edit the new file to include your details (e.g., name, ID), to display the date/time when a button is clicked, and to use a new CSS file
- Create and edit the CSS file to change the color of some text.



```
GNU nano 7.2                                12227071.html                                Modified
<html>
<head>
<title>Example web page</title>
</head>
<body>
<h1>Mohammad Bin Ali : ID : 12227071 </h1>
<button onclick="showDateTime()" style="color:blue"> Show Date and time </button>
<a href="index.html"> View Index.html</a>
<script>
function showDateTime() {
var dateTime = new Date();
alert(dateTime);
}
<h1>Example web page</h1>
</body>
</html>
```

^X Exit ^O Write Out ^W Where Is M-Q Previous ^K Cut ^C Location
^L Refresh ^R Read File ^\ Replace M-W Next ^U Paste ^_ Go To Line



Html Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
</head>
<body>
  <h1>Mohammad Bin Ali ID : 12227071 </h1>

  <button onclick="showDateTime()" style="color:blue">Show date and Time</button>
  <a href="index.html">View index.html</a>

  <script>
    function showDateTime() {
      var dateTime = new Date();
      alert(dateTime);
    }
  </script>
</body>
</html>
```

Task 2. Capture HTTP Packets

```
<button onclick="showDateTime()" style="color:blue"> Show Date and time </button>
<a href="index.html"> View Index.html</a>
<script>
function showDateTime() {
var dateTime = new Date();
alert(dateTime);
}
<h1>Example web page</h1>
</body>
</html>
```

```
root@OpenWrt:/srv/www# tcpdump -i eth0 -n -w http-12227071.pcap 'not tcp port 22'
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```
147 packets captured
147 packets received by filter
0 packets dropped by kernel
```

```
root@OpenWrt:/srv/www# _
```

```
root@OpenWrt:/srv/www# ls
12227071.html      http-12227071.pcap  index.html
root@OpenWrt:/srv/www# _
```

Windows 10 x64 - VMware Workstation

File Edit View VM Tabs Help

kali-linux-2022.4-vmware-am... X Kali Main X Windows 10 x64 X

Windows PowerShell

Ethernet adapter Bluetooth Network Connection:

Media State : Media disconnected
Connection-specific DNS Suffix . :
PS C:\Users\Loyd San> arp

Displays and modifies the IP-to-Physical address translation tables used by address resolution protocol (ARP).

ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -a [inet_addr] [-N if_addr] [-v]

-a Displays current ARP entries by interrogating the current protocol data. If inet_addr is specified, the IP and Physical addresses for only the specified computer are displayed. If more than one network interface uses ARP, entries for each ARP table are displayed.

-g Same as -a.

-v Displays current ARP entries in verbose mode. All invalid entries and entries on the loop-back interface will be shown.

inet_addr Specifies an internet address.

-N if_addr Displays the ARP entries for the network interface specified by if_addr.

-d Deletes the host specified by inet_addr. inet_addr may be wildcarded with * to delete all hosts.

-s Adds the host and associates the Internet address inet_addr with the Physical address eth_addr. The Physical address is given as 6 hexadecimal bytes separated by hyphens. The entry is permanent.

eth_addr Specifies a physical address.

if_addr If present, this specifies the Internet address of the interface whose address translation table should be modified. If not present, the first applicable interface will be used.

Example:

> arp -s 157.55.85.212 00-aa-00-62-c6-09 Adds a static entry.
> arp -a Displays the arp table.

PS C:\Users\Loyd San> arp -a

Interface: 192.168.119.129 --- 0x6	Internet Address	Physical Address	Type
	192.168.119.2	00-50-56-f1-1c-d5	dynamic
	192.168.119.254	00-50-56-e6-7c-a4	dynamic
	192.168.119.255	ff-ff-ff-ff-ff-ff	static
	224.0.0.22	01-00-5e-00-00-16	static
	224.0.0.251	01-00-5e-00-00-fb	static
	224.0.0.252	01-00-5e-00-00-fc	static
	239.255.255.250	01-00-5e-7f-ff-fa	static
	255.255.255.255	ff-ff-ff-ff-ff-ff	static

PS C:\Users\Loyd San>

Task 3 Analyze HTTP Packet Capture

a) For each HTTP request/response, provide a short explanation of: what triggered the request, what was requested and what was the response. For example: "The user clicked on the link ... which caused the browser to send a HTTP Request for /page.html. The server did not have that page so responded with ...".

Ans. Everytime we request some content on the web browser, it is sent via http - GET Method. Simple searching for the url, triggered the request. Entering the URL in the browser, i.e ip of our OPENWRT Machine, asks the server to serve the file requested, index.html is by default.

b) For the first HTTP request/response, list the five (5) address values that identify the host, transport protocol and application.

Host: This is the domain name or IP address of the web server to send requests to. Identifies the target host for the request.

Protocol: This specifies the transport protocol used for requests/responses. For HTTP, the protocol is usually "HTTP/1.1" or "HTTP/2".

Port: This identifies the port number used for requests/responses. For HTTP, the default port is 80 for unencrypted requests and 443 for encrypted requests (using HTTPS).

Method: It specifies the HTTP method used for the request. B. GET, POST, PUT, DELETE, etc. A method specifies the type of action the client wishes to perform on the server.

User agent: This identifies the application making the request. It usually contains information about the web browser or other client software being used. B. Version number and operating system.

c) When you clicked on the button to show the date and time, did your browser send a request to the web server? Why or why not?

No, the browser does not send a request to the web server to display the date and time when the button is clicked. This is because the date and time display functionality is implemented using client-side scripting, specifically JavaScript executed by the client-side browser. This script gets the current date and time from the user's device and manipulates an HTML document to display the date and time. Requests are therefore handled entirely on the client side and do not require communication with the web server.

- d) One of the HTTP request/responses was for your newly created web page (e.g., 12345678.html). Draw a packet diagram for the request, and include the following information: - Size, in Bytes, of each header and of the entire HTTP request
- Addresses included in each header and/or HTTP request

```

Frame 35: 469 bytes on wire (3752 bits), 469 bytes captured (3752 bits)
Ethernet II, Src: VMware_1e:e1:ca (00:0c:29:1e:e1:ca), Dst: VMware_f1:1c:d5 (08:00:0c:29:1f:1c:d5)
Internet Protocol Version 4, Src: 192.168.119.134, Dst: 117.239.91.117
Transmission Control Protocol, Src Port: 42670, Dst Port: 80, Seq: 1, Ack: 1,
Hypertext Transfer Protocol
  POST / HTTP/1.1\r\n
    Host: r3.o.lencr.org\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
    Content-Length: 85\r\n
    Connection: keep-alive\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    \r\n
    [Full request URI: http://r3.o.lencr.org/]
    [HTTP request 1/5]
    [Response in frame: 37]
    [Next request in frame: 39]
    File Data: 85 bytes
Online Certificate Status Protocol
  tbsRequest
    requestList: 1 item
      Request
        reqCert

```

```

Wireshark - Packet 35 - http-12227071.pcap
Frame 35: 469 bytes on wire (3752 bits), 469 bytes captured (3752 bits)
Ethernet II, Src: VMware_1e:e1:ca (00:0c:29:1e:e1:ca), Dst: VMware_f1:1c:d5 (08:00:0c:29:1f:1c:d5)
Internet Protocol Version 4, Src: 192.168.119.134, Dst: 117.239.91.117
Transmission Control Protocol, Src Port: 42670, Dst Port: 80, Seq: 1, Ack: 1,
Hypertext Transfer Protocol
  POST / HTTP/1.1\r\n
    Host: r3.o.lencr.org\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
    Content-Length: 85
    Connection: keep-alive
    Pragma: no-cache
    Cache-Control: no-cache
    \r\n
    [Full request URI: http://r3.o.lencr.org/]
    [HTTP request 1/5]
    [Response in frame: 37]
    [Next request in frame: 39]
    File Data: 85 bytes
Online Certificate Status Protocol
  tbsRequest
    requestList: 1 item
      Request
        reqCert

```

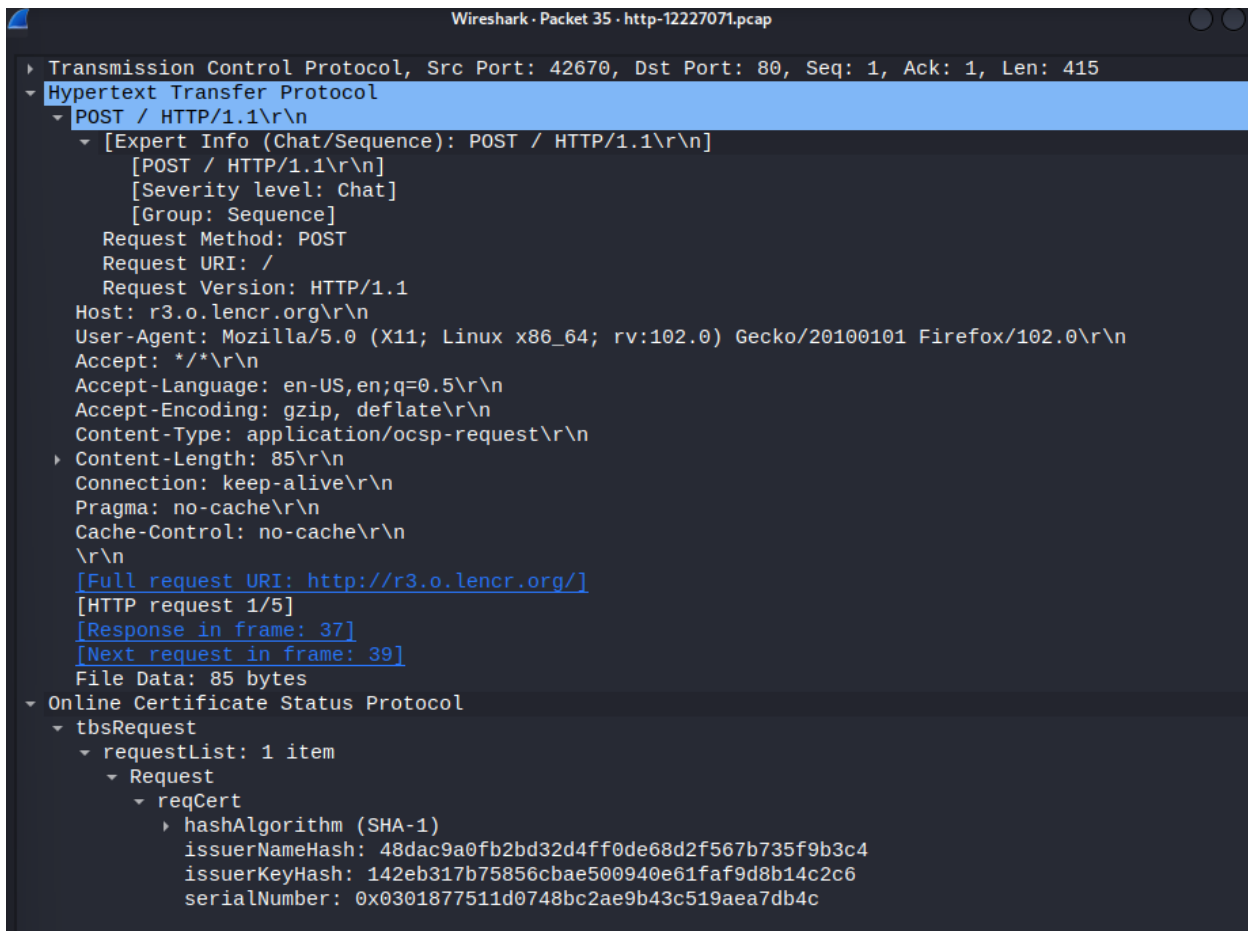
- e) For the HTTP request from part (d), what is the value of the referrer? What does it identify? How can web servers use this information?

A "referrer" is an HTTP header that specifies the URL of the web page the user was visiting before clicking a link or submitting a form to get to the current page. It is also sometimes spelled "Referer" (without the second "r"), a misspelling introduced into the HTTP specification for historical reasons.

A referrer can identify which page a user came from and provide useful information to the web server and her web application. For example, it can be used to track the effectiveness of advertising campaigns, see which search engines and other websites are sending traffic to a particular website. It can also be used to implement security measures. B. To prevent cross-site request forgery (CSRF) attacks.

Web servers can use referrer information in a variety of ways, depending on their particular needs and configuration. For example, referrers can be used by web analytics tools to track the source of website traffic, and e-commerce sites can use them to track the effectiveness of their marketing campaigns. Some web servers and web applications use referrer information for access control or other purposes by verifying that requests are from a trusted source (such as the same domain or subdomain as the current page). implement security measures.

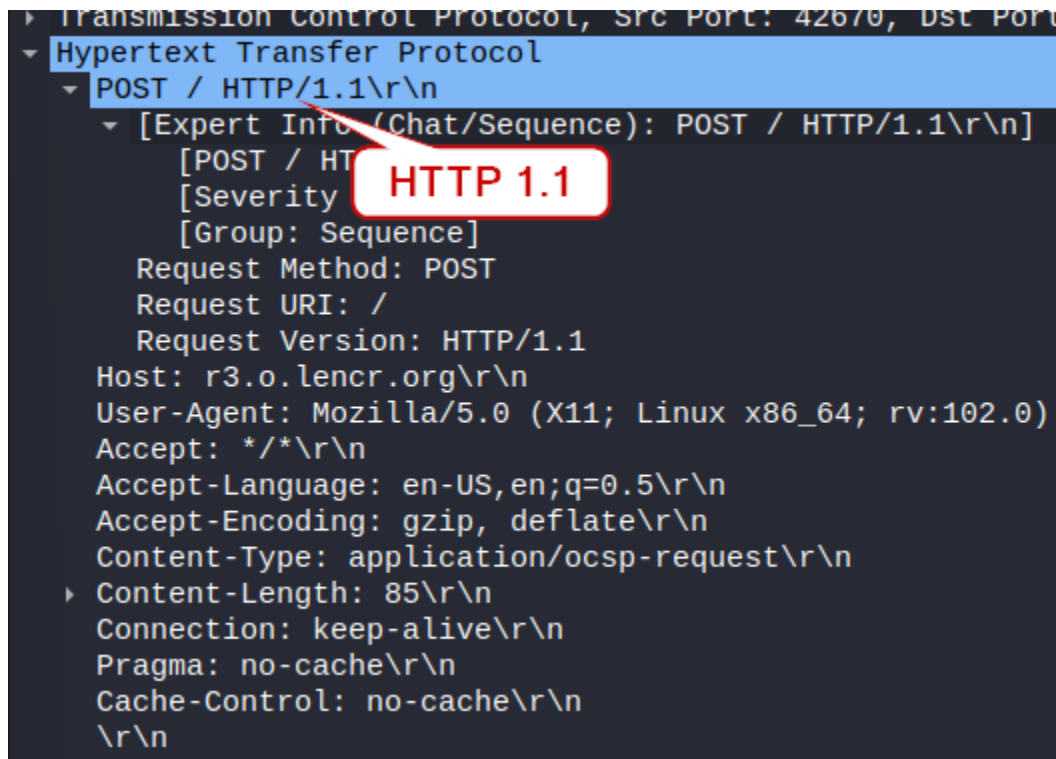
f) For the HTTP request from part (d), what information did the server learn about the web browser (e.g., name, version)?



```
Wireshark · Packet 35 · http-12227071.pcap
└─ Transmission Control Protocol, Src Port: 42670, Dst Port: 80, Seq: 1, Ack: 1, Len: 415
  └─ Hypertext Transfer Protocol
    └─ POST / HTTP/1.1\r\n
      └─ [Expert Info (Chat/Sequence): POST / HTTP/1.1\r\n]
        [POST / HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
        Request Method: POST
        Request URI: /
        Request Version: HTTP/1.1
        Host: r3.o.lencr.org\r\n
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0\r\n
        Accept: */*\r\n
        Accept-Language: en-US,en;q=0.5\r\n
        Accept-Encoding: gzip, deflate\r\n
        Content-Type: application/ocsp-request\r\n
        Content-Length: 85\r\n
        Connection: keep-alive\r\n
        Pragma: no-cache\r\n
        Cache-Control: no-cache\r\n
        \r\n
        [Full request URI: http://r3.o.lencr.org/]
        [HTTP request 1/5]
        [Response in frame: 37]
        [Next request in frame: 39]
        File Data: 85 bytes
    └─ Online Certificate Status Protocol
      └─ tbsRequest
        └─ requestList: 1 item
          └─ Request
            └─ reqCert
              └─ hashAlgorithm (SHA-1)
                issuerNameHash: 48dac9a0fb2bd32d4ff0de68d2f567b735f9b3c4
                issuerKeyHash: 142eb317b75856cbae500940e61faf9d8b14c2c6
                serialNumber: 0x0301877511d0748bc2ae9b43c519aea7db4c
```

g) What version of HTTP is used and what transport protocol is used?

HTTP 1.1



```
Transmission Control Protocol, Src Port: 42670, Dst Port: 80
Hypertext Transfer Protocol
  POST / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): POST / HTTP/1.1\r\n]
    [POST / HTTP/1.1]
    [Severity: Info]
    [Group: Sequence]
    Request Method: POST
    Request URI: /
    Request Version: HTTP/1.1
    Host: r3.o.lencr.org\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
    Content-Length: 85\r\n
    Connection: keep-alive\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    \r\n
```

h) A connection-oriented service involves setting up a connection before any data transfer, as well as acknowledgements that are used to provide reliability. Identify the packets involved in connection setup (e.g., the packet numbers). How long did it take between the start of connection setup and the data transfer starting?

Sync: The initiating device sends a SYN packet to the receiving device requesting connection establishment. This packet has a sequence number (SYN sequence number) and a randomly generated initial sequence number (ISN).

Sync confirmation: The receiving device acknowledges the connection establishment request by responding with a SYN-ACK packet. This packet contains a SYN sequence number and an acknowledgment number (ACK number) that is the SYN sequence number + 1. The package also contains a randomly generated ISN. confirmation:

The initiating device sends her ACK packet, acknowledging receipt of the SYN-ACK packet. This packet contains her ACK number which is her ISN + 1 on the receiving device.

Data transfer can begin as soon as a connection is established.

i) Identify the acknowledgements. When is an acknowledgement typically sent?

In connection-oriented protocols like TCP, the receiving device sends an acknowledgment (ACK) to confirm that it has received and successfully processed the data packet sent by the sending device. An acknowledgment is usually sent after one device receives a data packet from the other device. The acknowledgment contains the next sequence number that the receiving device expects to receive from the sending device. The sending device then uses this information to determine if packets that were not received by the receiving device should be resent.

The ACK packet has a sequence number that corresponds to the next expected sequence number of the data from the other side.

Task 4 View Your Cookies

Cookies are small text files that are stored on a user's device by a website when they visit it. They are commonly used to store information such as login details, user preferences, and shopping cart contents. Cookies can also store information about the user's browsing behavior, such as which pages they have visited or how long they spent on a particular website.

Cookies can be categorized based on their lifespan and the domain that they belong to. Session cookies are temporary cookies that are erased when the user closes their web browser. Persistent cookies, on the other hand, remain on the user's device until they expire or are manually deleted by the user.

First-party cookies are created by the website that the user is visiting, while third-party cookies are created by a domain other than the one that the user is currently visiting. Third-party cookies are often used for advertising and tracking purposes, which can potentially compromise the user's privacy.